ESD-TR-67-66
ESTI FILE COPY

ESD ACCESSION LIST
ESTI Call No. AL 55188
Copy No. 1

ESD-TR-67-66

ESD RECORD COPY
RETURN TO
SCIENTIFIC & TECHNICAL INFORMATION DIVISION
(ESTI), BUILDING 1211

MANAGEMENT HANDBOOK FOR THE ESTIMATION OF COMPUTER PROGRAMMING COSTS

Nelson, E. A.

31 October 1966

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

AD 648 750

Distribution of this document is unlimited.

(Prepared under Contract No. AF 19(628)-5166 by System Development Corporation, 2500 Colorado Ave., Santa Monica, California 90406)

AD0648750

LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
System Development Corporation
2500 Colorado Avenue
Santa Monica, California 90406

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b GROUP
N/A

3. REPORT TITLE
MANAGEMENT HANDBOOK FOR THE ESTIMATION OF COMPUTER PROGRAMMING COSTS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
None

5. AUTHOR(S) *(Last name, first name, initial)*
Nelson, E. A.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
31 October 1966	141	63

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 19(628)-5166	ESD-TR-67-66
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	None

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
AD 645 750	Directorate of Computers, Electronic Systems Division, Air Force Systems Command, USAF, L. G. Hanscom Field, Bedford, Mass. 01730

13. ABSTRACT

Guidelines are presented to help managers estimate the costs of computer programming. The guidelines summarize a statistical analysis of 169 computer programming efforts with equations to estimate man months, computer hours, and months elapsed, and also planning factors such as man months per thousand instructions. Opinions, rules of thumb, and experience data based upon literature search and experience supplement the statistical results. Forms with the guidelines are organized into six sections corresponding to a six-step division of the computer programming process. Advice is given on the integration of cost estimates into a cost analysis to justify and plan ADP projects.

DD FORM 1 JAN 64 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, *or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

GPO 886-551

FOREWORD

This report was prepared by the Programming Management Project at System Development Corporation for the Electronic Systems Division, Air Force Systems Command. The contents of the report and its organization as a Handbook are intended to supply a useful tool to managers for estimating the costs of computer programming. The contents summarize the results from a statistical analysis of costs and cost factors that characterize the design, code, and test work for 169 completed computer programming projects. These analytical results have been supplemented with opinions, rules of thumb, and experience data derived from the literature or based upon the experience of Project members that apply to other activities in computer programming such as information system integration test, as well as the computer program design, code, and test activity. The Handbook also presents advice on how cost estimates derived by means of the Handbook could be integrated into cost justification work and planning for ADP systems.

The Programming Management Project would like feedback (including constructive criticism and additional or better data) from readers on the usefulness of this document. To this end, a brief questionnaire designed to help evaluate the document has been included as the last page.

The statistical analysis that produced the numerical results in the section on computer program design, code, and test was conducted by the author (E. Nelson) and T. Fleishman and supported by H. Zagorski. V. LaBolle, Leader of the Programming Management Project, contributed to the integration and organization of the Handbook material.

This technical report has been reviewed and is approved.

George E. Vranesh

GEORGE E. VRANESH
1/Lt, USAF

CHARLES A. LAUSTRUP, Col, USAF
Chief, Computer and Display Division

ABSTRACT

Guidelines are presented to help managers estimate the costs of computer programming. The guidelines summarize a statistical analysis of 169 computer programming efforts with equations to estimate man months, computer hours, and months elapsed, and also planning factors such as man months per thousand instructions. Opinions, rules of thumb, and experience data based upon literature search and experience supplement the statistical results. Forms with the guidelines are organized into six sections corresponding to a six-step division of the computer programming process. Advice is given on the integration of cost estimates into a cost analysis to justify and plan ADP projects.

CONTENTS

Foreword		ii
Abstract		iii
List of Figures		vi
List of Tables		vii
I.	Introduction	1
	Purpose	2
	Scope	4
	Background and Sources of Data	4
	Cost Estimation Methods	5
	Approach of the Handbook	7
	Interpretation	11
II.	Using the Handbook	13
	Structure of the Handbook	13
	Preliminary Cost Evaluation Analysis	19
	Supplementing the Handbook	29
III.	Preliminary Planning and Cost Evaluation	33
	Activity Definition	33
	Computer Programming Cost Factors	36
	Additional Comments on Selected Cost Factors	38
	Planning Factors	39
IV.	Information System Analysis and Design	41
	Activity Definition	41
	Computer Programming Cost Factors	45
	Additional Comments on Selected Cost Factors	48
	Planning Factors	49

V. Computer Program Design, Code, and Test 51
Activity Definition 51
Computer Programming Cost Factors 54
Additional Comments on Selected Cost Factors 60
Planning Factors 65
Computer Programming Cost-Estimating Equations 74

VI. Information Processing System Integration Test 91
Activity Definition 91
Computer Programming Cost Factors 93
Planning Factors 96

VII. Information System Installation and Turnover 97
Activity Definition 97
Computer Programming Cost Factors 100
Additional Comments on Selected Cost Factors 104
Planning Factors 105

VIII. Computer Program Maintenance 107
Activity Definition 107
Computer Programming Cost Factors 110
Additional Comments on Selected Cost Factors 114
Planning Factors 115

Glossary A. Definitions of Variables 119
Glossary B. Definitions of Symbols 132
Glossary C. Use of Terms 133
Glossary D. Definition for the Code to Rate the Impact of Cost Factors 135

References 137

LIST OF ILLUSTRATIONS

FIGURE		
1	Computer Programming Project Cycle	6
2	Recycling the Computer Programming Process Steps	9
3	Sample Activity Definition Format	14
4	Sample Computer Programming Cost Factors	15
5	Sample Planning Factor Format	17
6	Sample Estimating Equations Format	18
7	Sample Cost Justification Form	21
8	Sample Project Description Form	24
9	Sample Budget/Schedule Work Sheet	26
10	Sample Budget/Schedule Summary	28

LIST OF TABLES

TABLE		
I.	Activity Definition--Preliminary Planning and Cost Evaluation	33
II.	Computer Programming Cost Factors--Preliminary Planning and Cost Evaluation	36
III.	Planning Factors--Estimation of Tasks Within Activity	39
IV.	Activity Definition--Information Processing System Analysis and Design	41
V.	Computer Programming Cost Factors--Information Processing System Analysis and Design	45
VI.	Planning Factors--Estimation of Tasks Within Activity	49
VII.	Activity Definition for Computer Program Design, Code, and Test	51
VIII.	Computer Programming Cost Factors--Computer Program Design, Code, and Test	54
IX.	Planning Factors--Quotes Without Comment	65
X.	Planning Factors--Resource Rate per 1000 SOURCE Instructions	66
XI.	Planning Factors--Resource Rate per 1000 OBJECT Instructions	67
XII.	Planning Factors--Total Pages of Documentation per 1000 OBJECT Instructions	68
XIII.	Planning Factors--Man Months Expenditure Rates	69
XIV.	Planning Factors--Estimation of Tasks Within Activity: Unit Cost	70
XV.	Planning Factors--Estimation of Tasks Within Activity: Percent of Other Item	71
XVI.	Planning Factors--Program Development Computer Time	72
XVII.	Planning Factors--Estimating Computer Hours from Man Months	73

TABLE

XVIII.	Comparison of Estimating Equation Characteristics	76
XIX.	Equation for Estimating Man Months: Total Sample	77
XX.	Equation for Estimating Computer Hours: Total Sample	78
XXI.	Equation for Estimating Months Elapsed: Total Sample	79
XXII.	Equation for Estimating Man Months: Medium Computer	80
XXIII.	Equation for Estimating Months Elapsed: Medium Computer	81
XXIV.	Equation for Estimating Man Months: Large Computer	82
XXV.	Equation for Estimating Months Elapsed: Large Computer	83
XXVI.	Equation for Estimating Computer Hours: Other Subsample	84
XXVII.	Equation for Estimating Computer Hours: Utility Subsample	85
XXVIII.	Equation for Estimating Months Elapsed: POL Subsample	86
XXIX.	Equation for Estimating Man Months: MOL Subsample	87
XXX.	Activity Definition--Information Processing System Integration Test	91
XXXI.	Computer Programming Cost Factors--Information Processing System Integration Test	93
XXXII.	Planning Factors--Costing System Integration Test	96
XXXIII.	Activity Definition--Information Processing System Installation and Turnover	97
XXXIV.	Computer Programming Cost Factors	100
XXXV.	Planning Factors--Costing Demonstration Test	105
XXXVI.	Planning Factors--Estimation of Tasks Within Activity	106
XXXVII.	Activity Definition--Computer Program Maintenance	107
XXXVIII.	Computer Programming Cost Factors--Computer Program Maintenance	110

TABLE

XXXIX.	Planning Factors--Estimating Number of Maintenance Programmers	115
XL.	Planning Factors--Estimating Program Maintenance Computer Hours: Unit Cost	116
XLI.	Planning Factors--Estimating Program Maintenance Computer Hours: Unit Cost	117
XLII.	Planning Factors--Estimating Program Maintenance Computer Hours: Percent of Other Item	118

SECTION I

INTRODUCTION

This Handbook is an organized collection of material intended to help managers estimate the cost of computer program development. Specifically, both quantitative and qualitative guidelines are given for estimating the resources, i.e., man months, computer hours, and months elapsed, to be used in conducting the six activities that are assumed in this Handbook to constitute the computer programming process. This division of the process forms the basis for the organization of the Handbook and includes the following sections: Preliminary Planning and Cost Evaluation; Information System Analysis and Design; Computer Program Design, Code, and Test; Information System Integration Test; Information System Installation and Turnover; and Computer Program Maintenance. In addition to a brief description of the activity in terms of tasks, inputs, and outputs, each of these sections includes a list of cost factors together with some indication of their influence on costs and planning factors such as production rates, e.g., man months per 1000 instructions and comparisons between the costs of the particular activity to total process costs. Reflecting the infant state of the art in planning for computer programming, the guidelines presented for most of the programming activities are mostly qualitative, based upon the authors' experience or upon data and opinions found in the technical literature. An exception to the general dearth of quantitative data in the Handbook, the section on Computer Program Design, Code, and Test presents the results of a statistical analysis of numerical data on costs and cost factors that describe 169 completed programming projects. This section, in addition to the quantitative planning factors, includes equations to estimate the resources needed for conducting the program design, code, and test activity. Several sets of equations were derived--based upon data from entire sample as well as data from subsamples to identify the type of programming languages used (machine-oriented language versus procedure-oriented language), the type of application (business, scientific, software, and other), and the type of computer (small, medium, and large--based upon cost).

It is assumed that one use of the cost estimates for computer programming is to help managers decide whether to proceed with the implementation of plans for the computer program being costed. To aid the manager in organizing the data for a cost evaluation, the section on Using the Handbook contains examples of forms for recording cost estimates to compare alternatives. The section on Preliminary Planning and Cost Evaluation contains information on the factors that affect the cost of planning and making the estimate itself.

This Handbook does not compare and evaluate the various guidelines presented. Use of all the applicable guidelines for any particular job would undoubtedly yield a number of estimates with a wide variation. Even the results derived by statistical analysis do not provide highly accurate estimating relationships. Therefore, this Handbook should be interpreted as an early effort to collect

and systematize some of the available knowledge on cost estimation for computer programming. As such, it should be regarded by the user as a supplement to managerial judgment rather than a replacement for it.

Further, we recommend that, wherever possible, the users of the Handbook supplement the numerical data provided by recording data on costs and cost factors on their own computer programming work and by analyzing these data to obtain improved guidelines.

The remainder of this Introduction clarifies the purpose of the Handbook, defines the scope of the material, identifies the sources of data, describes the methods of estimation, introduces the division of the computer programming process, and presents the caveats on data accuracy and reliability. The next section, Using the Handbook, explains (1) the various forms and tables that display the data, (2) how to use the data in preparing a cost evaluation, and (3) the ways in which the user could supplement the Handbook.

1. Purpose. IBM's expected $200 million investment in software for the System 360 computer series and their much-publicized difficulties in delivering many of the computer programs dramatically highlight the lack of effective tools for controlling and planning computer programming (26). Managers at IBM are not alone; underestimates for costs and lead times are the rule rather than the exception in the teeming new industry of computer programming. One reason for this situation is that relatively little has been done to record, systematize, and generalize technical management experience. Meanwhile, the number of new machine installations and new applications continues to grow rapidly, demanding new managers--often inexperienced. At the same time, more and more experienced managers are swallowed up in the ambitious frontier-breaking ventures such as large military or space systems with major ADP subsystems, large integrated time-sharing networks for government and industry, and development of computer programs (software) for new computers. The void of organized knowledge and formal material for the technical management of computer programming dictates that most learning now and in the future probably will be accomplished by personnel obtaining actual experience in the field.

But there are some efforts under way to accumulate, integrate, and convert experience into useful guidelines. Among these efforts is the Programming Management Project at System Development Corporation (SDC), whose members developed this Handbook under a contract with the Air Force Electronic Systems Division, Deputy for Engineering and Technology, Directorate of Computers.

The major purpose is to begin to provide the operating manager of computer programming projects with a methodology and the data that can help him forecast the resources required and incorporate these estimates into cost evaluation studies, broader project plans, and cost control systems. To these ends, the available data are organized in consistent formats, and guidelines are given for how to use these data in preparing a cost evaluation for a proposed computer programming effort.

A second purpose of the Handbook is to encourage the manager to collect and analyze data on costs and cost factors from his own operations. The formats themselves provide both a way and an incentive for an estimator to supplement the data presented. These additions to the material are needed for several reasons:

- As indicated earlier, the only section in the Handbook with a representative sample of numerical data is that on Computer Program Design, Code, and Test. To improve the usefulness of the other sections, numerical data should be collected and analyzed to derive planning factors and estimating equations.

- Even the estimating relationships provided in the section on Computer Program Design, Code, and Test, although based upon statistical analysis of a reasonably-sized sample, still have large standard errors. Therefore, we expect new estimating relationships based upon collection of local data and their analysis in an individual organization might be more accurate than those derived in our statistical analysis because many of the sources of cost variation would be eliminated. That is, many of the factors identified as variables in this Handbook such as the type of computer application and personnel would be held relatively constant in a particular organization or installation.

- Finally, new data will be needed to accurately reflect the changing ADP technology. Change is the predominant characteristic of the world of computers, and in this, computers with features such as multiprocessors with logarithmically increasing speeds have been the forerunners. Applications that feature networks with automatic inputs and outputs and time-sharing capabilities highlight the transition in computer configurations. New programming techniques such as data management systems, query languages, programming languages, and compilers are also a significant part of the change. Such changes have a marked impact on the economics of data processing generally, and consequently on the estimation of programming costs. The principles of estimation will not change; the basic ways of arriving at a cost estimate today will be the same. But the data, and the cost factors and equations developed from them, will change, i.e., the material in this Handbook will become obsolete.

Therefore, a Handbook such as this, to remain useful, must be dynamic. It requires addition to make it more complete and accurate, and continual updating to reflect changes such as new compilers, machines, training methods, or programming languages. And so, this volume, designed as a Handbook, may take on more of the characteristics of a workbook--periodically updated and continually improved.

2. Scope. We confined the scope of this Handbook to the estimation of the costs in terms of resources required in the computer programming process and the organization of these cost estimates for management review. The broader but related problems of planning and control are not directly addressed.

We also do not address narrower subjects that are directly related to costs such as selection, training, and appraisal of computer programmers (35, 46, 49) equipment costs,[1] and the associated question of purchase versus lease (9, 54) Prices to convert the estimates of basic resource units (man months, computer hours, elapsed time) to dollar costs are not provided.[2]

This Handbook is intended as a concise reference covering the particular subject of computer programming cost estimation. It is not a treatise on the subject, but a compilation of facts and authoritative opinion. Wherever possible, we used charts or tables in place of prose. For this reason, most of the prose is found in this Introduction and in the next section on how to use the Handbook.

3. Background and Sources of Data. The Handbook summarizes the results of a major task in the Programming Management Project at SDC--to derive equations for estimating the costs of the computer program design, code, and test activity in computer programming. To conduct this task, Project members used a questionnaire to collect numerical data that measure costs and cost factors for completed programming efforts and subsequently analyzed these data using statistical methods. These analyses were done in cycles; each succeeding cycle, aimed at improving earlier results, corresponded to the collection of new numerical data and their subsequent analysis. In the first cycle, data for 27 programming efforts (data points) completed at SDC were analyzed and the results were reported in the fall of 1964 (19). The work in the second cycle, an analysis of a total of 74 data points also representing SDC programming work, was reported in detail in the fall of 1965 (62), and summarized and extended in the spring of 1966 (53). The third cycle, which included analysis of 169 data points,[3] 69 of which were programmed at SDC and 100 at Air Force and industrial programming organizations, provides most of the quantitative guidelines found in the section on Computer Program Design, Code, and Test.

---

[1] For prices of various items of equipment, physical characteristics and operational characteristics including the performance of certain configurations on benchmark problems, see (5) There have also been some attempts to develop cost estimating relationships for new computer hardware for special applications (29).

[2] For salary ranges for different job categories, see (17).

[3] This data base, and the statistical methods used in the analysis, are summarized in (20).

In addition to the results of the statistical analysis, the contents of this Handbook were obtained from two other sources of readily available material:

a. The general published literature in the files of the Programming Management Project. Primarily, periodicals were reviewed. An extensive search of all the available literature could not be made. The references used are listed at the end of this volume.

b. The technical knowledge and accumulated experience of members of the Programming Management Project at SDC. All statements that are not specifically referenced should be considered the best judgment of the Project members.

4. Cost Estimation Methods. The data and guidelines in the Handbook can be used in various ways to make a cost estimate. In using these methods, an analyst must infer some sort of relationship between the unknown future costs and accumulated past experience (39). There are essentially four methods by which this can be done:

a. Specific analogy, where costs for a new item are estimated by using the known costs for a similar item produced earlier.

b. Unit price, where the cost of a new item is estimated by the product of the number of units to be delivered in the new item (e.g., number of instructions) and previously determined cost per unit.

c. Percent of other item, where the cost of a new item is estimated as a predetermined percent of the cost of another item, e.g., the cost of Computer Program Design, Code, and Test might be a fixed fraction of total computer programming costs.

} planning factors

d. Parametric equations, where the cost of the new item is estimated from an equation which is a function of various characteristics of the requirements for the item resources expected to be used and working conditions.

Each of these methods is comparative; each is dependent upon an applicable data base from past experience, each assumes that the past is prologue. To estimate a particular job, each method may be used alone, or in combination with the others. They also may be applied at various levels of aggregation; e.g., to estimate computer programming costs, the total computer program may be estimated as a whole, or broken up into components, such as steps in the programming process. All estimates of costs, no matter how subjective they may appear to be, are actually based on one or more of the above four methods.

In the sections of this Handbook, corresponding to six activities or steps in computer programming, we present data to enable the user to make estimates using the last three methods cited: unit price, percent of other cost, and parametric

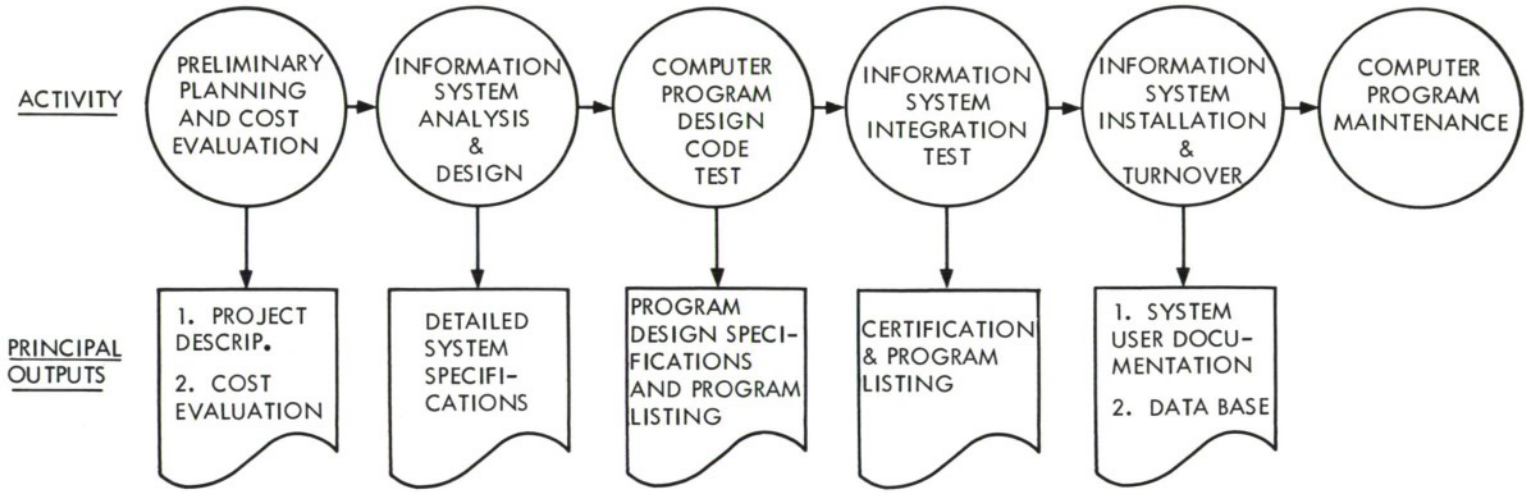


FIGURE 1

COMPUTER PROGRAMMING PROJECT CYCLE

equations. The first two of these are called Planning Factors in the text. To use the specific analogy technique, some effort is under way to develop an index of ADP system development efforts in the Air Force (23). Ideally such an index would identify and help retrieve histories of completed projects that were similar to a proposed ADP system. The costs of the completed efforts could then be used to estimate the new project by the specific analogy method.

5. Approach of the Handbook. In the Handbook, we have divided the programming process into distinct steps, or activities, for planning and estimating purposes. From the technical manager's viewpoint, there are two reasons for breaking up the programming process into steps. Different steps may represent fundamentally different kinds of jobs with consequently different cost implications, such as different types of personnel, tasks, and/or locations. Also, if the completion of a step can be a clearly defined and identifiable event with a definitive end product, these events constitute milestones useful for control.

The steps making up the computer programming process, or project cycle, are assumed to be:

a. Preliminary Planning and Cost Evaluation

b. Information System Analysis and Design

c. Computer Program Design, Code and Test (production)

d. Information System Integration Test

e. Information System Installation and Turnover

f. Computer Program Maintenance

These steps, with their principal outputs, are illustrated in Figure 1; a more detailed description of the tasks and outputs of each step is provided in succeeding sections. In this Handbook, we regard an information (processing) system as possibly containing many components or subsystems such as computers, computer programs, communications, displays, and operational procedures, all designed or tailored and used to work together to help an organization perform one or more missions. A large (high cost) computer programming effort usually corresponds to the development of a complex information system in which most, if not all, subsystems and components would be new or changed. In this case, all the subsystem developers, for example, the computer contractor, would carry on a set of activities similar to those listed above. In the simplest kind of system development, from the viewpoint of the computer programming organization, all of the components or subsystems would remain relatively fixed, e.g., same computer, and the system analysis and design would be minimal, with the remainder of the activities confined to the computer program as a single component.

Using the earlier reasoning that identification of more tasks and monitoring of these will help improve management, even more steps or divisions would be desirable for control and planning. Previous work by the Programming Management Project and others on the planning problem in computer programming has used even finer divisions for the process (6, 18, 30). However, the absence of cost data would make further division of the process in this Handbook a useless exercise. Many times, for planning purposes, the tasks of Program Design, Coding, and Testing are separated, but in the Handbook they are grouped together as a single activity because the numerical data used in the analysis were gathered in that way. The other five activities corresponding to the remaining divisions were identified for several reasons:

- To stimulate recognition of a complete spectrum of computer programming work in planning that involves cost forecasts. Even without large amounts of numerical data, a manager may avoid underestimates if he recognizes that each of the activities may involve a major expenditure of manpower or lead time.

- To stimulate planning and cost comparison work in computer programming by identifying a specific activity for such work as part of the process.

- To recognize and acknowledge the trend toward larger integrated information systems by identifying the potentially costly activities that follow computer program design, code, and test, e.g., system integration test. The need to identify, plan for, and cost such activities is seen more clearly for large information systems because the major subsystems and components in a large information system are more evident, e.g. costly. One expects to have a major system design effort and system integration testing. In reality the tasks that constitute such activities are almost always performed (although usually subsumed in the planning) on even the smallest computer programming job that results in an operational computer program.

The steps in the programming process imply a sequence in which the completion of work within each step is prerequisite to the start of work on the following step. In practice, however, this principle may not be apparent, because a project may be divided into subprojects wherein work proceeds at different rates for various subprojects. Also, work completed on a process step may have to be repeated because of later developments in the process; for example, the failure of a program to pass an integration test will require repeating some part of the program design, code, and test step, or the system analysis and design step, or both. This recycling within the programming process, illustrated in Figure 2, is widely recognized as a major factor in both the magnitude of total costs and the uncertainty of predictions for programming costs.

In conducting the analysis that led to the results in the section on Computer Program Design, Code, and Test, we assumed that computer programming, regardless of application and the types of resources used, has certain characteristics

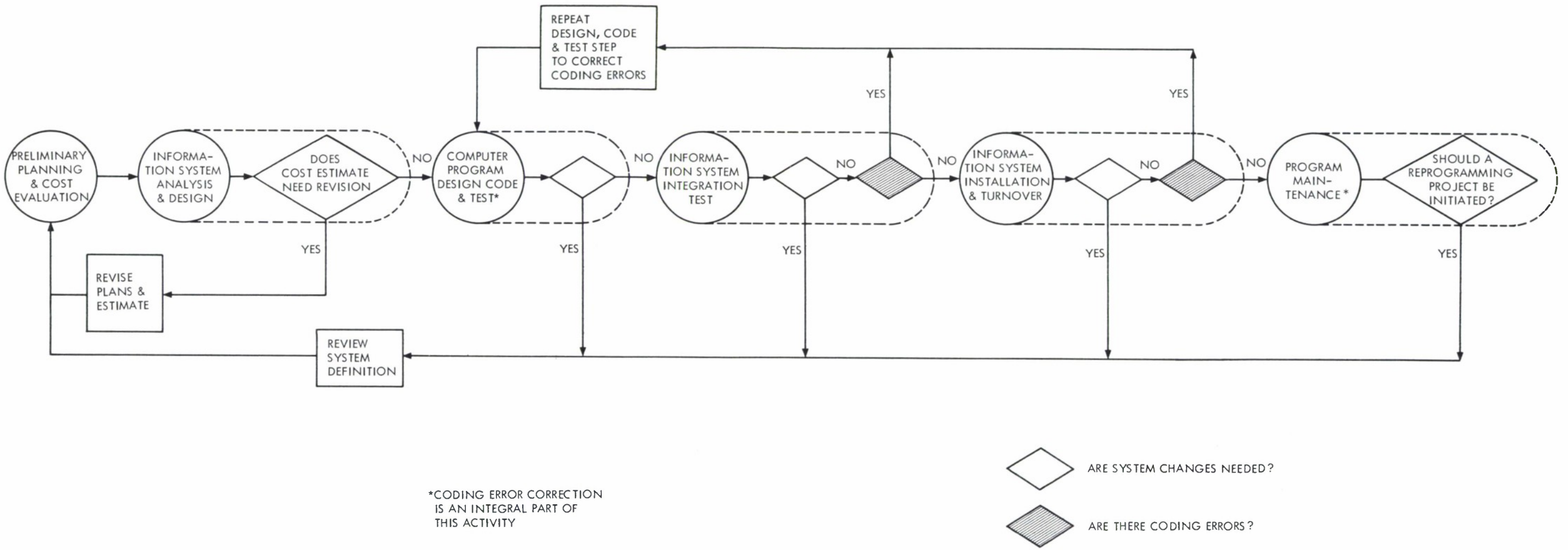


FIGURE 2

RECYCLING THE COMPUTER PROGRAMMING PROCESS STEPS

(page 10 blank)

that can be generalized, and that variation in costs from job to job can be accounted for by the variation in a selected set of these characteristics, or cost factors. Thus, the primary costs, manpower and machine usage, can be considered as dependent variables that can be expressed as a function of these cost factors (independent variables). An implicit assumption is that the programs whose cost data form the data base are representative of those of any user of the Handbook. This, of course, will not usually be true.

To try to tailor the analysis so that more useful results could be derived for specific programming organizations, we divided the data into classes to distinguish a number of different kinds of computer programs, such as business, scientific, and programs written for large computers. Even more work of this type is desirable. However, the size of our data base does not permit statistical analysis of finer divisions that might be useful for characterizing the computer programming work in a particular organization.

6. Interpretation. All the data used from both the statistical analysis and the literature were data of opportunity, i.e., we took what we were able to get in the time available. Hard data on the costs of computer programming or, more generally, automatic data processing economics are scarce commodities both in computer programming organizations and in the published literature. Few numerical data are recorded; fewer yet are recorded under "controlled" conditions, and still fewer are suitable for generalization to other situations. In the rapidly multiplying field of computer programming, diverse opinions on the costs and benefits of techniques and applications are rampant. This wide variation in the literature is matched by the variation in the numerical data that we used in the statistical analysis. The respondents to the questionnaire were under no obligation to assure completeness and accuracy even when data were readily available. Because they were suspect, some of the data collected were rejected prior to the analysis. But even those data used in the analysis are likely to have a variation in reliability, which we believe has been smoothed by the use of statistical techniques applied to a sample of reasonable size.

The user of this Handbook is likely to discover conflicting data from other sources, and obtain different estimates using the various material supplied by the Handbook. These conflicts characterize the state of the art in cost estimation and indicate the limitations of the data base available to this project. Specific limitations of the data base used in the preparation of the Handbook include inaccuracies in reporting data (including guesswork by respondents), misinterpretations of questions by respondents, inclusion of costs that were not a part of the program design, code, and test step, and inappropriate definition of data items or cost factors on the questionnaire. We assume that the reader will recognize the limitations of the data in applying them to his specific problems.

In this sense, the Handbook can be considered a first step to supplement (not to replace) judgment by managers. Because the predominant trend in planning for computer programming has been to underestimate the costs, the best advice that can be offered at this time is to lean toward the conservative (realistic) estimates for costs.

The following breakdown lists the number of different types of computer programs supplied by each source that contributed to the data base. This list may be of some assistance to the reader in making inferences about the applicability of the Handbook data to his specific application.

DISTRIBUTION OF DATA BY PROGRAMMING APPLICATION

			Total Data Points	Type of Program			
				Business	Scientific	Computer Software	Other
Govt		U. S. Air Force*	38	26	10	2	
Industry	Computer Software Research and Development	Company A	6	3		3	
		Company B	1			1	
		Company C	1	1		0	
		Company D	69	17	12	5	35
	Computer Hardware and Aerospace	Company E	2		2		
		Company F	3	2	1		
		Company G	21	19	2		
		Company H	28	11		17	
		Total	169	79	27	28	35

*Note: Data represent 14 separate USAF organizations.

SECTION II

USING THE HANDBOOK

To explain the use of the Handbook, this section describes (1) the structure of the remaining sections (corresponding to the steps in the programming process), including the formats for the material, and (2) a way in which the estimates that result from use of the guidelines can be introduced into the planning and an evaluation of costs for a proposed effort in computer programming. We conclude the section with some brief advice on how to supplement the Handbook based upon our experience in gathering and analyzing data.

1. Structure of the Handbook. Exclusive of this section and the Introduction, the Handbook is divided into six basic sections--one for each step of the programming process for which costing information is supplied. These data for each activity in the programming process are presented in a uniform manner. Each section contains the following kinds of formatted material:

- An activity definition that includes a list of major tasks, inputs, and outputs for the particular programming step.
- Major cost factors--a list, along with some indication of the impact of these factors, followed by discussion of those factors for which we have additional information.
- Planning factors that can be used with unit price and percent-of-other-item estimating methods described in the Introduction.
- Estimating equations currently available only for the Program Design, Code, and Test activity).

In any section, a symbol, a replica of Figure 1, appears at the top of the first form of each of four types and indicates, by shading, which activity or step is being addressed. The forms and the types of information these contain are described in more detail below.

a. Activity Definition. The Activity Definition for each step in the computer programming process is presented as shown in Figure 3 with the following items:

(1) Description. The description is a concise general statement of what the activity is. In addition, for readers that are involved in electronic system development for the Air Force, we have included information that relates the sequence of activities to the planning and control for computer programming that is reflected in the Air Force guidelines on Systems Management that will be found in the Air Force Systems Command Manuals in the 375 series (6, 61).

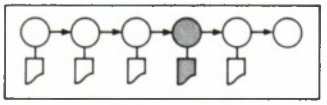


TABLE XXX: ACTIVITY DEFINITION

ACTIVITY:	INFORMATION PROCESSING SYSTEM INTEGRATION TEST
DESCRIPTION:	This activity covers all work necessary to test the performance of the computer program within ths total system at the operational facility under realistic ("live") operating conditions. (AFSCM 375 context: This activity occurs in the Acquisition Phase and is equivalent to Category II testing.)
TASKS:	Conduct Test. Within the requirements of the plans for program testing, conduct a sequence of tests of the total operational system, receiving actual data from and transmitting actual data to other subsystems and components that constitute the system. Analysis of Test Results. Determine if the system meets specifications, and study operations for any evidence of potsntial difficultiss. Coordinate with other subsystem and component developers to isolate sources of poor system performance. Initiate Modifications to Computer Programs. Correct errors in programs and associated documentation. Design and implement feasible changes to meet specified performance requirements. This involves work in the earlier activities. Documentation of Test Results. Prepare appropriate compliance documentation to certify successful tests. If problem areas arise, document the evidence for use in the modification process. Identify potential improvements that could be made to the total system.

FIGURE 3

SAMPLE ACTIVITY DEFINITION FORMAT

(2) Tasks. To detail each activity, a checklist of the major tasks that may be part of the activity is provided. Not all computer programming efforts will involve all of the tasks listed.

(3) Inputs and Outputs. The inputs and outputs describe the information required to perform work on the specified activity, and the products of the specified activity, respectively.

The documents such as specifications and statement listings, cited as outputs of each activity, are the direct products of the programming process that are needed to perform the next step or to aid in the operation of the computer program by the user. We excluded items prepared for technical management such as activity reports, and project control and cost reports. Each output listed for a process step provides not only a basis for understanding the activity, but also the basis for control of the programming project by using a deliverable item with a completion date as a milestone.

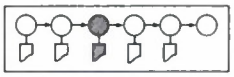


TABLE VIII — COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_1 - Vagueness of design requirements definition	+2B	+2B	+2B	86		
X_2 - Innovation required	+2B	+2B	+2B	80, 83, 86		
X_3 - Lack of knowledge of operational requirements	+2B	+2B	+2B	77, 85		
X_4 - Number of organizational users	1B	1B	+2B	80, 81		
X_5 - Number of ADP centers	+3	+2	+2	87		
X_6 - Complexity of program interface with other systems	+2B	+2	+2B	80, 82		
X_7 - Response time requirements	+4A	+4A	+4A		13, 21	
X_8 - Stability of design	+4	+3	+4	77-81, 83, 85-87		
X_9 - On-line requirements	1B	1B	+2B		13, 21, 85, 44 44	
X_{78} - Complexity of system interfaces	+	+	+			
X_{83} - Degree of system change expected during operations	+	+	+			
X_{84} - Number of functions in the system	+	+	+			
X_{85} - Number of system components	+	+	+			

*NOTE: IMPACT IS INDICATED BY:

CORRELATION
4 = HIGH
3 = MODERATE
2 = LOW
1 = INDETERMINATE

SION
+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SION = NO PRESUMED DIRECTION

OTHER CONSIDERATIONS
A = HIGH CORRELATION, BUT ALSO CROSS-CORRELATION
B = STRONG INTUITIVE APPEAL

(FOR DISCUSSION OF CODING, SEE GLOSSARY B)

FIGURE 4

SAMPLE COMPUTER PROGRAMMING COST FACTORS FORMAT

b. Major Cost Factors. The next type of form found in each section (see Figure 4, "Computer Programming Cost Factors") lists cost factors, i.e., those variables that affect the expenditure of resources in the particular activity being addressed. Four types of resources are used in computer programming:

(1) Manpower measured in units such as man months.

(2) Computer usage, measured in units of time such as hours of use for main-frame of a computer.

(3) Elapsed time, measured in units of time such as months.

(4) Other ADP costs, such as expenditures for supplies expressed in dollars.

The completed form, as in Figure 4, shows how each factor influences the first three resources--manpower, computer usage, and elapsed time. Other costs, such as those for supplies, are not discussed in this Handbook. The form in Figure 4 contains seven columns. The first column contains the names and numbers of cost factors that are defined more completely in Glossary A. The

next three columns, labeled "Impact on Indicated Resources," contain information on the influence of the factor upon the three major costs--manpower, computer usage, and elapsed time.

In the section on the Computer Program Design, Code, and Test activity, the entries in these three columns contain a sign (plus or minus) indicating whether or not a factor increases or decreases costs, and a number (based upon a statistic--the correlation coefficient) that indicates the amount of the impact. In the other sections for the other five activities, since no statistical analysis was conducted, only the direction (sign) of the influence is shown in these three columns.

The next three columns in Figure 4 are labeled "Reference." The first column, "Handbook Page," contains numbers of pages in the Handbook where additional information on a particular factor can be found, e.g., in the discussions immediately following the list or in the planning factors. The next two columns, labeled "Descriptive" and "Analytical," contain the numbers of references that discuss the particular cost factor. Descriptive references do not contain numerical data, while Analytical references do.

These lists of factors (and the discussions that follow them) provide at least two benefits to the user of this Handbook. First, the estimator can interpret and evaluate his own situation vis-à-vis the planning factors and the estimation equations presented later. That is, he can consider the relative impact of the various cost factors on his specific problem to help him decide on which estimating relationship to use. Secondly, the list of factors that affect cost can be used as a checklist for setting up a system to control costs by managerial attention to the causal relationships involved. For example, the importance of stability of design as a factor in total costs argues for thorough initial planning and a management policy that minimizes changes in system requirements.

The cost factors are not necessarily independent, although this would be desirable. On the contrary, the statistical analysis on the numerical data for Computer Program Design, Code, and Test showed that many of the factors are highly intercorrelated (e.g., X_{58}--machine access time and X_{59}--machine add time, X_{10}--total object instructions and X_{17}--number of conditional branches). There may also be considerable difference of opinion as to precisely how a factor should be defined or measured. But the purpose of this Handbook is not to resolve these matters; instead, it is designed to provide a useful list of factors and definitions, and a format such that the list of factors and definitions can be added to or modified to meet the particular needs of different users.

c. Planning Factors. The third type of information found in the remaining sections of the Handbook is labeled "Planning Factors." The data on forms permit the user to use the unit price and percent-of-other-item methods for estimating costs. These methods require that the user have information on the

number of items needed or information on the cost of the other items. Even when one or the other of these two pieces of information is available, the user must still use judgment to decide which planning factor to use because there may be several that could apply.

As indicated earlier, most of the numerical data in the Handbook appear in the section on Computer Program Design, Code, and Test. An example of a completed Planning Factor form that appears in this section is shown in Figure 5. This example presents unit price data--man months, computer hours, and months elapsed per 1000 object instructions for various subsamples such as language type, application type, and computer type.

This use of number of instructions as a normalizing parameter, i.e., instruction as the unit in the unit-price data, is common throughout the Handbook particularly in the section on Computer Program Design, Code, and Test. In using this device we do not distinguish among different computers in terms of logical power of the instructions. Also in the section on Computer Program Design, Code, and Test we do not account for the characteristic inefficiency of Procedure-Oriented Languages and their associated compilers. Because of this inefficiency, the use of a POL yields, on the average, more machine-language instructions than use of Machine-Oriented Language (symbolic or assembly language) and an assembler, to perform the same logic or solve the same problem.

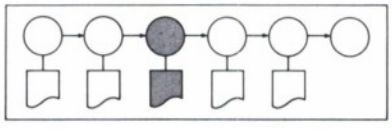


TABLE XI — PLANNING FACTORS

TYPE: UNIT COST — SUBJECT: RESOURCE RATE PER 1000 OBJECT INSTRUCTIONS

TYPE OF COMPUTER PROGRAM	NO. OF DATA POINTS	RESOURCE COST RATE: MAN MONTHS $\frac{Y_1}{X_1} \times 10^3$ MAX	MIN	σ	MEDIAN	MEAN	COMPUTER HOURS $\frac{Y_2}{X_2} \times 10^3$ MAX	MIN	σ	MEDIAN	MEAN	ELAPSED TIME (MO.) $\frac{Y_3}{X_3} \times 10^3$ MAX	MIN	σ	MEDIAN	MEAN
MOL	123	100.00	.14	10.18	4.00	5.89	294.04	.05	42.75	15.00	29.52	40.00	.06	5.81	1.33	3.55
POL	46	9.49	.07	2.61	1.16	2.13	52.50	.30	13.74	2.86	9.76	10.43	.06	3.73	.98	2.30
		$P < .05$					$P < .01$					N.S.				
FORTRAN	8	9.49	.11	3.88	.97	2.75	50.69	.31	18.02	2.68	10.25	18.43	.17	6.92	1.14	4.50
JOVIAL	15	7.60	.66	2.11	2.50	3.07	58.50	.77	15.06	16.67	17.73	8.38	.47	1.86	1.18	1.87
COBOL	12	9.33	.07	2.53	.49	1.25	30.33	.30	10.70	1.90	5.87	7.33	.06	1.98	.41	1.08
OTHER POL	11	4.10	.12	1.57	.57	1.36	14.49	.31	3.86	2.43	3.45	11.27	.06	3.56	.74	2.64
		N.S.					$P < .05$					N.S.				
BUSINESS	79	38.82	.07	5.11	1.54	3.13	80.00	.23	17.93	3.09	11.95	18.94	.06	4.05	1.11	2.87
SCIENTIFIC	27	12.00	.14	2.90	3.14	3.55	140.00	.05	26.52	4.83	17.78	18.43	.11	4.28	1.18	2.93
UTILITY	28	100.00	.49	18.83	3.28	9.79	294.04	1.06	68.30	30.18	57.36	21.00	.10	4.03	1.03	2.03
OTHER	35	17.65	.66	4.18	4.20	5.69	129.03	3.86	28.42	20.98	30.00	40.00	.24	8.41	2.00	5.14
		$P < .01$					$P < .01$					$P < .25$				
LARGE COMPUTER	105	100.00	.07	10.51	3.17	5.50	294.04	.05	42.47	12.50	26.72	40.00	.06	5.99	1.00	3.08
MEDIUM COMPUTER	53	16.67	.12	3.09	2.54	3.19	177.77	.31	30.22	5.20	17.57	18.94	.06	3.92	2.00	3.19
SMALL COMPUTER	11	38.82	.93	10.81	4.00	6.97	80.00	1.09	24.51	34.69	31.14	14.93	.64	4.78	2.98	4.66
		N.S.					N.S.					N.S.				
TOTAL SAMPLE	169	100.00	.07	8.94	2.93	4.87	294.04	.05	38.16	10.44	24.14	40.00	.06	5.34	1.22	3.21

NOTES:

1) P = Probability of erroneously concluding that the population means are different (e.g., $p < .01$ indicates that if one concludes that the population means are different, the probability of error is less than 1 out of 100). If $p > .25$, it is denoted by N.S. (not significant) in this table.

2) All of the planning factors above are based on raw data that has not been winsorized (i.e., extreme values reduced) to prevent distortion by large values.

FIGURE 5

SAMPLE PLANNING FACTORS FORMAT

In addition to the form shown in Figure 5, other forms will be used to present the planning factors, e.g., plots of data to exhibit the percent-of-other-item cost data. Also, in some instances where data are available, planning factors for tasks or subtasks that comprise one of the major activities in the computer programming process will be listed.

d. Estimating Equations. As indicated earlier, estimating equations are only available for the Computer Program Design, Code and Test activity. These results of the current analysis conducted by the Programming Management Project include equations for estimating manpower, computer usage, and months elapsed that are derived from and apply to the entire sample as well as various subsamples. The results of the subsample analyses were included only if the equations derived were more accurate (i.e., had some smaller errors) than the equations for the entire sample. The format in which the equations are presented, for example, as in Figure 6, contains the following:

(1) The equation itself in terms of numbered variables, X_i. The variables are defined on the fold-out (page 89).

(2) In addition to the identification by table number, a heading that describes the sample under Data Base and the resources being estimated under Resource.

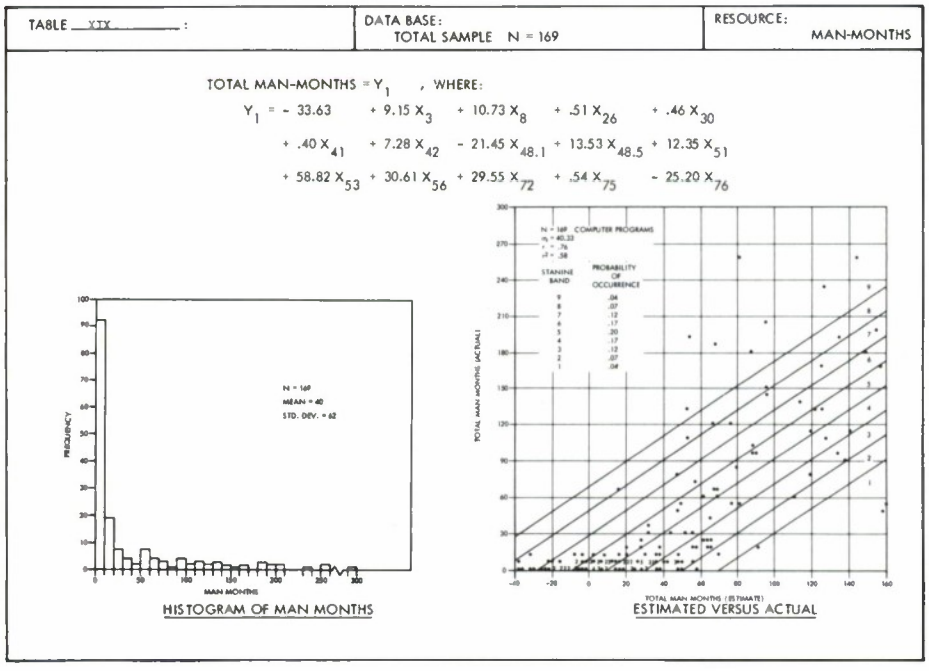


FIGURE 6

SAMPLE ESTIMATING EQUATIONS FORMAT

(3) A frequency distribution in the form of a histogram for the data that were collected for the indicated resource.

(4) A plot of estimated values using the indicated equation on the factors for each data point against its actual values for the particular resource in the data base. This plot also contains Stanine Bands to portray the statistical confidence in the derived estimating equation.

On plots such as the Estimated versus Actual Man Months in Figure 6, the confidence levels are indicated by the inserted table, Stanine Band Number versus Probability Values. These bands can be interpreted as follows (16): Suppose an estimate of 80 man months has been calculated. By reading on the vertical line for this value in Figure 6, we see that the probability (or chance) that the actual value will fall between 68 and 92 man months (the values of boundaries on the Number 5 Stanine Band) is found in the table as .20 (or 20 out of 100 programming jobs). Using the same estimate of 80 man months, the probability that the actual value will fall between 53 and 107 (the lower and upper boundaries for the Number 4 and Number 6 Stanine Bands respectively that bracket the Number 5 Stanine Band) is .54, the sum of the probabilities shown in the table for the Numbers 4, 5, and 6 Stanine Bands. Note that the bands are symmetric around a 45-degree line and their width is equal to one-half the standard error of estimate given in the upper left-hand corner of the plot. The standard error depends upon the sample size and the power and efficiency of the predictors used in deriving the equation. Another sample may widen or narrow the Stanine Bands leading to changes in the estimating precision.

2. The Preliminary Cost Evaluation Analysis. The Preliminary Planning and Cost Evaluation activity, the first step in the computer programming process, includes the actual estimation of costs. Section III in the Handbook describes the activity and planning factors for estimating the resource cost of the step itself. This step is unique; it not only triggers the work on the other steps of the programming process, but interfaces with other management activities. The output of the cost evaluation step may well determine whether or not the project should be continued.

To show the relationship of cost estimation to planning or, in particular, how the cost estimates made by using the material in this Handbook can be used in a cost evaluation for a proposed computer programming effort as part of the development of a data processing system, we discuss the Preliminary Planning and Cost Evaluation work in terms of four forms:

- Figure 7, A Sample Cost Justification Form that provides a way to compare costs for a proposed computer program as part of an ADP system with the costs for an existing system (automatic or manual) that performs the same data processing functions.

- Figure 8, A Sample Project Description Form that would contain the numerical values for cost factors such as system and resource characteristics to describe the proposed computer programming project. These factors are the inputs needed to use the cost-estimating relationships in the Handbook.

- Figure 9, A Sample Budget/Schedule Work Sheet that could be used to record the allocation of resources (both in resource units, e.g., man months and dollars) for each activity over several intervals of time as well as to record a total for each activity (summed over the time interval) and a grand total for the entire project.

- Figure 10, An ADP Project Budget/Schedule Summary that can be used to record the costs from Figure 9 by fiscal year intervals and to combine the computer programming costs with equipment costs. To relate it to Air Force and DOD planning, the form provides for division of the costs into Research and Development (R and D), Investment, and Operating costs, the categories used by the Air Force in preparing a long-term plan (42). A provision for discounting costs is also made in the form.

These four forms (Figures 7 through 10) should be viewed as suggestions for ways to record the evaluation of computer programming costs and to integrate these costs into long-range plans.

a. The Cost-Justification Format. The cost-justification procedure outlined here consists of a comparison of the total costs (development and operating) expected for the proposed system (or project) with the operating costs of the present system. This comparison can be conveniently made on a form such as illustrated in Figure 7, a data processing project cost evaluation summary. This form is aimed at comparing costs for an ADP system or project that requires computer programming work as part of the development and operations. The right-hand side of Figure 7 contains summary costs for the proposed system, the left-hand side, the summary costs of the existing system. Modifications of this form might include provision for several alternatives in design for the proposed system. The meaning of each item of Figure 7 is as follows:

Item 1, Assumed Life of the Project. Operating costs accrue continuously, or at various time intervals, for the life of the project. If the value is to exceed the cost, ultimately, the nonrecurring costs of procuring a system must be amortized by savings in future operating costs. The total estimated life of the project (measured in years) during which these savings may be realized is recorded in Item 1.

DATA PROCESSING PROJECT

COST EVALUATION SUMMARY

REMARKS:

1. ASSUMED LIFE OF PROPOSED PROJECT: _____ YRS.

2. PROPOSED PROJECT NON-RECURRING COSTS

ADP COSTS:

- A. PRELIM. ANAL. _____*
- B. SYST. SPEC. _____*
- C. DES. CODE & DEBUG _____*
- D. SYST. TEST _____*
- E. SYST. TURNOVER _____
- F. EQUIP. PURCHASE _____
- G. EQUIP. INSTALL. _____

NON-ADP COSTS _____

SUB TOTAL _____

3. PRESENT SYSTEM OPERATING COSTS

ADP COSTS:

- A. HARDWARE MAINT. _____ /YR.
- B. HARDWARE RENTAL _____ /YR.
- C. HARDWARE OPERATION _____ /YR.
- D. MAINT. PROGRAMMING _____ /YR.
- E. OTHER _____ /YR.

NON-ADP COSTS _____ /YR.

SUB TOTAL _____ PER YR.

4. PROPOSED PROJECT OPERATING COSTS

ADP COSTS:

- A. HARDWARE MAINT. _____ /YR.
- B. HARDWARE RENTAL _____ /YR.
- C. HARDWARE OPERATION _____ /YR.
- D. MAINT. PROGRAMMING _____* /YR.
- E. OTHER _____ /YR.

NON-ADP COSTS _____ /YR.

SUB TOTAL _____ PER YR.

5. TOTAL PRESENT SYSTEM COST _____

6. TOTAL PROPOSED SYSTEM COST _____

7. DISCOUNTED TOTAL COST (RATE= ; PERIOD=) _____

8. DISCOUNTED TOTAL COSTS (RATE= ; PERIOD=) _____

9. JUSTIFICATION:

- A. TOTAL PRESENT SYSTEM COST (ITEM 6) - TOTAL PROPOSED SYSTEM COST (ITEM 7) = _____
- B. ADDED $ BENEFITS FROM PROPOSED SYSTEM = _____

*NOTE: DETERMINED FROM DATA IN THIS PROGRAMMING MANAGEMENT HANDBOOK

C. TOTAL VALUE = [ ]

FIGURE 7

SAMPLE COST JUSTIFICATION FORM

Item 2, Proposed Project Nonrecurring Costs. Nonrecurring costs are those associated with the procurement and installation of a new system. This Handbook was specifically intended to help in making estimates for Items 2a through 2e in Figure 7, that is, the costs of the various activities in the computer programming sequence. The costs of purchase (Item 2f) and installation (Item 2g) of any new equipment are also identified as non-recurring costs for the project under consideration. We also include in this category one-time costs such as the costs of personnel training, purchase and installation of non-ADP equipment or material or project costs such as management procedures.

Item 3, Present System Operating Costs. These costs (Items 3a through 3e) for an ADP system include hardware maintenance, rental and operation, as well as program maintenance. If the present data processing system is entirely manual, these annual costs would be summarized under the "Non-ADP Costs" heading. If the proposed data processing system is a completely new function, with no existing system to replace, the present system costs would be zero. In this case, economic justification would be based entirely on the proposed system costs and Item 9b in Figure 7, added benefits of the proposed system measured in dollars. Since investment, that is, the non-recurring, costs for the present system are sunk costs, they are not considered (4). That is, the funds have been committed and are not eligible for allocation in making this particular decision.

Item 4, Proposed Project Operating Costs. The types of items in annual operating costs for a proposed ADP system are the same as for the present system (Item 3) and include computer hardware maintenance (Item 4a), computer rental (Item 4b) or payments if equipment is not to be purchased outright (in the latter case for purchase cost, the down payment would be included in Item 2f), and ADP operation costs; ADP operation costs (Item 4c) include the costs of operating personnel and power. Annual maintenance programming costs (Item 4d) are those associated with this function as defined in Section VIII of this Handbook. Other annual ADP operating costs (Item 4e) include supplies, keypunch, and all other recurring ADP related work.

Non-ADP operating costs cover the annual costs of associated manual systems, the proposed project's share of overhead, direct supervision, and all other recurring costs not previously accounted for.

Item 5, Total Present System Costs. This item in Figure 7 represents the sum of all the annual costs in Item 3, multiplied by assumed life (in years) of the proposed system.

Item 6, Total Proposed System Costs. This item in Figure 7 represents the sum of all annual costs in Item 4 multiplied by the assumed life (in years) of the proposed system, plus the sum of all nonrecurring costs in Item 2.

Item 7, Discounted Total Cost of Present System. The present value of the current system cost (Item 5) discounted at the indicated annual interest rate ("the value of money") over the time period for the assumed life of the system is summed and entered here. If discounting is not desired (i.e., if the rate = 0), this value is the same as entered in Item 5.[4]

Item 8, Discounted Total Cost of Proposed System. The present value of the proposed system cost (Item 6), discounted at the indicated interest rate over the time period for the assumed life cycle, is summed and entered here. The interest rate and time period for discounting proposed system costs to their present value must, of course, be the same as used for the present system. If discounting is not desired, this value is the same as entered in Item 6.[4]

Item 9, Justification. The ultimate justification of a proposed ADP project depends on the expected savings and/or additional dollar benefits expected. This calculation may be readily made in Item 9 of Figure 7. Note that expected savings (i.e., cost of old system less cost of new system) may be negative, and indeed they must be when the proposal is not a replacement for an existing procedure; the entire burden of acceptance then rests with the expected ability of the proposed system to generate new dollar returns. The procedure for calculation of expected economic returns for proposed ADP projects is a subject beyond the scope of this Handbook.

The final decision to proceed with a project may depend upon many more considerations than merely obtaining a positive number for the total economic value in Item 9c of Figure 7. For example, the absolute magnitude of the proposed system cost (Item 6) may exceed available resources. Another important consideration is the percent of expected total value (Item 9c) to total proposed system cost (Item 6); this is because of the uncertainty attached to any estimate, and the possibility of higher-than-expected costs wiping out anticipated returns.

b. Defining System Characteristics. The cost-evaluation summary illustration in Figure 7 provides an overview of the kinds of estimates required to justify the cost of an ADP project. To develop actual numbers for the costs of a proposed project, the cost factor values for the project being estimated must be specified. Figure 8 provides a suggested format for this.

Items 1 through 9 and 24 through 26 of Figure 8 are self-explanatory. Items 10 through 23 should contain the names and the numerical values for the important cost factors (system and resource characteristics) that are known or can be estimated at this early stage. To determine what is important, the user should

---

[4] For a discussion of discounting in return on investment analysis, see (4). The time phasing of expenditures may be worked out on forms such as Figure 9.

PROGRAMMING PROJECT DESCRIPTION				
1. PROJECT TITLE:				
2. EXPECTED START DATE:	3. EXPECTED COMPLETE DATE:	4. RFP #:	5. CONTRACT #:	6. SECURITY CLASSIFICATION
7. PROJECT MONITOR:		8. PRIME CONTRACTOR:		
9. DESCRIPTION OF PROPOSED SYSTEM: WHAT IT DOES, HOW IT DOES IT.				

SYSTEM CHARACTERISTICS		
10.	11.	
12.	13.	
14.	15.	
16.	17.	
18.	19.	
20.	21.	
22.	23.	
24. PREPARED BY:	25. ORGANIZATION:	26. CURRENT DATE:

FIGURE 8

SAMPLE PROJECT DESCRIPTION FORM

read through the other sections in the Handbook as well as Glossary A, the Definitions of Cost Factors, and use his judgment. For example, these entries could include the independent variables such as those used in the estimating equation for Program Design, Code, and Test or the basis for the selection of particular planning factors from the range of values available in the Handbook.

The Sample Project Description Form, Figure 8, is one way that such information may be summarized and presented. Other formats and data items may be desired, or even contractually required, in certain instances (2). For example, subordinate forms could be prepared that describe the work to perform each of the various steps. These forms could list factors that influence the cost of the particular step as well as their estimated values.

c. Making a Numerical Estimate. To prepare the numerical estimate for use in the Cost Evaluation Summary, Figure 7, information such as that recorded in Figure 8, Programming Project Description, is used with the guidelines in the Handbook in the following manner. Figure 9, The Programming Project Budget/Schedule Work Sheet, is a suggested form in which only the programming costs could be entered.

(1) Select the appropriate planning factor or estimating equation, for each step in the computer programming process. This selection is guided by the system characteristics and project description (as in Figure 8) determined in the previous step. The selection may also be influenced by the desire of the estimator to provide a margin of safety to cover the many uncertainties involved. When statistical measures are available that show the expected spread of the estimated costs, such as standard deviations for planning factors or Stanine bands for equations, these yield important insights into the statistical uncertainties inherent in the data base.

(2) Calculate total man months, computer hours, and the other resources (e.g., supplies) required for each separate step in the programming process, using the planning factors and/or equations selected above. In Figure 9, these entries would be made in column 11.

(3) Distribute the estimated totals over an appropriate time span. The work sheet, Figure 9, divided into periods appropriate for the size and time span expected for the project, provides for recording these estimates. This preparation of the plan for performance is guided by the following considerations:

(a) The normal sequence of steps in the computer programming process. If the computer program for the project is divided into subprograms with different schedules, additional work sheets similar to Figure 9 would be desirable for scheduling and budgeting each subprogram. Of course, the schedules for all subprograms should intersect at the end of the Computer Program Design, Code, and Test activity when the total program system is tested as a unit.

PROGRAMMING PROJECT
BUDGET / SCHEDULE WORK SHEET

LIN	ITEM	PERIOD 1		PERIOD 2		PERIOD 3		PERIOD 4		PERIOD 5		TOTAL		SOURCE OF ENTRY OR REMARKS
		UNITS	$	UNITS	$	UNITS	$	UNITS	$	UNITS	$	UNITS	$	
		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	
1	PRELIMINARY ANALYSIS, MM													
2	CH													
3	OTHER													
4	SUB TOTAL		—		—		—		—		—		—	
5	SYSTEM SPEC., MM													
6	CH													
7	OTHER													
8	SUB TOTAL		—		—		—		—		—		—	
9	DESIGN, CODE & DEBUG, MM													
10	CH													
11	OTHER													
12	SUB TOTAL		—		—		—		—		—		—	
13	SYSTEM TEST MM													
14	CH													
15	OTHER													
16	SUB TOTAL		—		—		—		—		—		—	
17	SYSTEM TURNOVER MM													
18	CH													
19	OTHER													
20	SUB TOTAL		—		—		—		—		—		—	
21	MAINT. PROG. MM													
22	CH													
23	OTHER													
24	SUB TOTAL		—		—		—		—		—		—	
25	TOTAL PROGRAMMING COSTS, $													

FIGURE 9

SAMPLE BUDGET/SCHEDULE WORK SHEET

(b) Any constraint on the total time available for completion. Delivery dates, dictated by external requirements, may determine the total number of personnel working on the project from the total man months required and the delivery dates specified.

(c) Any constraint on the resources such as manpower or computer hours available per time period for the project.

(d) Preferred time spans for the activities based on the technical factors involved. The Planning Factors and Estimating Equations in this Handbook will help the user estimate the amount of elapsed time needed in various situations.

(4) Convert the resource units (man months, computer hours) into dollar values. That is, the resource units are multiplied by the dollar cost per unit applicable at the facility in question. Figure 9, the Budget/Schedule Work Sheet, provides for entries that result from these calculations.

(5) Check for reasonableness of estimate. For example, make a direct comparison of the costs of the project in question with the historical costs of other similar projects. This technique is a variation of the specific analogy method of cost estimating described earlier. In certain circumstances, it may also be desirable to obtain the advice of experts or consultants, e.g., by securing an outside bid on portions of the work (39).

(6) Integrate the dollar costs estimated for computer programming such as those recorded in Figure 9, the Budget/Schedule Work Sheet, into an overall budget for the ADP project that includes equipment and other costs. Figure 10 is an example of a form in which such information may be summarized; data for the items that are steps in the programming process are derived from Figure 9. The items in Figure 10 are divided into Research and Development (R&D), Investment, and Operating categories; this division corresponds to the format used by USAF in long-range planning. Figure 10 also provides for entering the results of a discounting calculation.

(7) Incorporate dollar estimates into the cost-evaluation summary such as Figure 7.

This estimating process will usually be repeated through several iterations during the progress of a programming project (for example, see Figure 2). During the first step of the programming process described in Section III of the Handbook, the first preliminary estimate is made. If the decision is made to proceed, the information system analysis and design step will usually provide revisions to some of the original estimating assumptions, thus requiring a new cost estimate. Also, changes in the information system design and/or specifications may be made at various stages in the programming process, at the instigation of management or the customer, or because of the results of testing; such changes may make new cost estimates advisable. And finally,

	LINE	ITEM	YR. 1 (1)	YR. 2 (2)	YR. 3 (3)	YR. 4 (4)	YR. 5 (5)	TOTAL (6)	SOURCE OF ENTRY
		ADP PROJECT BUDGET / SCHEDULE SUMMARY (DOLLARS)							
R & D		ADP COSTS:							
	1	PRELIMINARY ANALYSIS						*	
	2	SYSTEM SPEC.						*	
	3	DESIGN, CODE, & DEBUG						*	
	4	SYSTEM TEST						*	
	5	R & D SUB-TOTAL							
INVESTMENT		ADP COSTS:							
	6	SYSTEM TURNOVER						*	
	7	EQUIPMENT PURCHASE							
	8	EQUIPMENT INSTALLATION							
	9	NON ADP COSTS							
	10	INVESTMENT SUB-TOTAL							
OPERATING		ADP COSTS:							
	11	COMPUTER MAINTENANCE							
	12	COMPUTER RENTAL							
	13	COMPUTER OPERATIONS							
	14	MAINTENANCE PROGRAMMING						*	
	15	OTHER ADP COSTS							
	16	NON ADP COSTS:							
	17	OPERATIONS SUB-TOTAL							
	18	TOTAL DIRECT							
	19	G & A							
	20	FEE OR PROFIT							
	21	GRAND TOTAL							
	22	DIS. RATE _____: DISCOUNT FACT	1.00						
	23	DISCOUNTED TOTALS							

* NOTE: ONLY STARRED TOTALS (*) COMPUTED FROM MATERIAL IN THIS HANDBOOK.

FIGURE 10

SAMPLE BUDGET/SCHEDULE SUMMARY

the management control system for the programming project may indicate overruns and/or slipping schedules, requiring a revision of estimates or a modification of objectives or both.

3. Supplementing the Handbook. As indicated earlier, one purpose of the Handbook is to encourage managers to adopt an analytical viewpoint and to collect and analyze data on their own computer programming operations. Specific suggestions on the types of data that can be collected can be found in the Computer Programming Cost Factor forms in Sections III through VIII. Glossary A defines these variables in more detail. Techniques and procedures for analyzing the data are described in earlier reports by the Programming Management Project at SDC (19, 20, 62). These documents contain references that provide even more details on the statistical techniques.

Analyses of local operations have certain potential advantages and disadvantages. These pros and cons are discussed below along with some recommendations:

a. Advantages

- A manager responsible for a single programming organization can greatly reduce the number of variables (cost factors) considered in this Handbook by selecting or defining those that he feels apply to his particular situation, e.g., type of job, equipment, personnel resources. This reduction, in turn, simplifies and hence reduces the cost of the analyses of the data.

- A manager can more readily identify those cost factors susceptible to control in his own operation and can actually control them. As a result, he can reduce the variation in costs and improve his planning.

- Also, he can identify and measure the impact of the cost factors that cannot be controlled (usually variables that characterize requirements) and thereby improve his pricing techniques.

- With an accumulation of some data that provide local standards, a manager can more easily identify and account for the cost differences associated with changes in programming techniques, equipment, personnel, personnel training, organization, and technical management policies.

b. Disadvantages

- On the other hand, local data collection and analysis may not be effective because sufficient data cannot be collected within one organization to derive local standards for cost that will be useful for control and planning. Although the number of factors can be reduced, thus permitting a smaller sample to be used in the analysis,

too drastic a reduction could purge factors that actually account for variation in local costs and the analysis may yield poor estimating relationships.

- Using more factors as a basis for data collection requires a larger sample (more data points corresponding to computer programming efforts) to derive equations. To accumulate data on an adequate sample may take a long time in one organization because a single programming effort may take several months to complete.

- Time and money are needed to collect and analyze data. Further, some discipline is required to "reserve" the time of the personnel responsible to assure that the collected data are actually analyzed.

c. Recommendations. A compromise might be to have uniform data collected locally and forwarded to some central organization where these data would be analyzed. Useful results would then be fed back to contributing organizations. The cost analysis that led to the results in Section V, Computer Program Design, Code, and Test, was an approximation to this approach. However, these data on completed programming projects were not usually based upon records made while the projects were under way, and some of the data were thus unreliable. Another problem in data accuracy stemmed from the absence of face-to-face interviews in collecting data; terms used in the mail-out questionnaire were misinterpreted. As a result, additional time was used to check and correct the data, and even then, some data were discarded.

To anticipate such problems and promote the useful collection of data, we recommend the following:

- Despite the disadvantages, do collect and analyze data. Even if strong statistics are not realized quickly, insight into the distribution of costs will be gained.

- Collect the data in terms of computer programming products, i.e., deliverable end items. The usual question on pricing and budgets is, "What will it cost to do a particular job?"

- Define the costs and cost factors as precisely as possible in terms that are common to other organizations. This permits the manager to use (1) data and cost-estimating relationships from other programming organizations or cost studies, and (2) any cost standards that may be developed by Federal Government agencies.

- Collect data while the projects are under way, by activities (as in this Handbook) or by selected milestones. If the data are not recorded as they are generated, personnel must rely on recall or search through related records, reducing the reliability of the data and adding to the cost.

- Identify a central repository and person who is responsible for the storing of the data (an organization with many programming jobs could use automatic data processing techniques) to avoid loss of parts of the data.

- Record initial estimates of the costs for which standards or estimating relationships are desired, and feedback from analyses that include comparison of early estimates with actuals. This will close the loop and provide the most immediate benefit to the responsible personnel in making improved estimates. Also, these personnel may be able to identify reasons for differences between actuals and estimates; these can be used to create new cost factors.

- Identify and record reasons for revisions to estimates during a project. Again, these are potential cost factors.

- Do not start a data collection, unless the intent and resources are to be made available for analyses and feedback. If the feedback loop is not completed, costs expended are largely wasted and personnel may be left with impressions that would cause them to resist future data collection efforts.

- Test any proposed scheme for data collection on a small sample of projects to assure that the scheme is feasible and understandable by the range of personnel who will have to provide source data.

- Try to forecast the changing technology in any data collection effort, and be prepared to make changes to accommodate any unanticipated new developments.

- Finally, try to keep some data on the cost of the data collection and analysis work itself, and compare these costs with the benefits derived.

(page 32 blank)

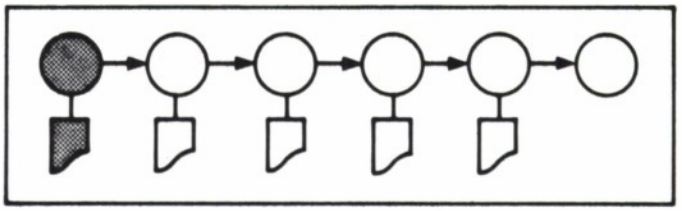


TABLE I:	ACTIVITY DEFINITION
ACTIVITY:	PRELIMINARY PLANNING AND COST EVALUATION
DESCRIPTION:	This activity consists of the economic feasibility study for the proposed program. Based on a statement of the user's requirements, an estimate is made of the manpower, computer time, elapsed time, or other resources required for the project. Using these estimates, a summary project plan and a cost versus benefits comparison are prepared. No more analysis of the proposed information system is done during this activity than is absolutely necessary for cost estimation and preliminary planning purposes.
TASKS:	Determine System Requirements and Characteristics. Study user's requirements. Make a preliminary analysis of the environment (organization, hardware, data base requirements), and any similar existing computer programs to determine the values of the parameters necessary for a cost estimation. Select Appropriate Planning Factors. Based on the characteristics of the proposed system, select the appropriate planning factors and/or cost estimating equations to be used. Sources for these factors include this Handbook, other published material and historical records of the installation making the estimate. Calculate Computer Programming Costs. For each step in the programming process, calculate the man months, computer hours, and other resources required for the proposed project using the previously determined planning factors. Convert units to dollar equivalents.

TABLE I: ACTIVITY DEFINITION (CONT'D.)

Determine Project Costs Other than Programming. Prepare estimates of other costs associated with the proposed project. These costs include: equipment purchase and/or installation; equipment maintenance, rental, and operation; data base conversion; overhead costs; personnel training; supplies.

Check Reasonableness of Estimates. Compare estimates made by several different methods. Compare totals with those of similar projects. Obtain expert opinion.

Prepare Summary Budget Plan. Develop project schedules by allocating cost expenditures over time. Prepare summary (i.e., by step in the programming process within major subproject breakdown structure), Gantt and/or PERT charts, and budgets for performing organizations.

Determine Costs of Existing System. Determine the costs of accomplishing the objectives of the proposed project with the current system. Establish a dollar value for gains of the proposed project that are over and above the benefits expected from replacing a present system.

Determine Economics Feasibility of Proposed System. Compare the costs and benefits of the proposed system to those of the existing system. Determine if the absolute magnitude of the costs involved is within the limits of resources of the organization. Determine if the required resources, such as programmers, will be available within the established schedules.

PRINCIPAL INPUTS:

User's requirements and environment.

System requirements and environment.

Planning factors and cost-estimating relationships.

Unit dollar costs of resources for the facilities involved.

Total resources available to the project at the facilities involved.

TABLE I: ACTIVITY DEFINITION (CONT'D.)

PRINCIPAL OUTPUTS:	Project description.
	Resource-unit, e.g., man months and dollar-cost estimates.
	Tentative schedules.
	Cost justification project evaluation summary.

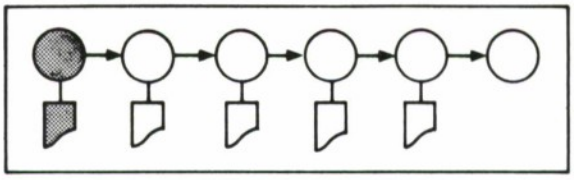


TABLE II: COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_1 - Vagueness of design requirements definition	+		+			
X_3 - Lack of knowledge of operational requirements	+		+			
X_{78} - Complexity of system interfaces	+		+			
X_{84} - Number of functions in system	+		+	38		
Program Design & Production						
X_{13} - Total source instructions expected (i.e., expected size of program)	+		+	38		
X_{41} - Number of subprograms	+		+	38		
$X_{45.1}$ - Internal documentation expected	+		+			
X_{46} - External documentation required	+		+			
$X_{47.1}$ - Total number of external document types expected	+		+			
$X_{47.3}$ - Total number of internal document types expected	+		+			
X_{48} - Type of program						

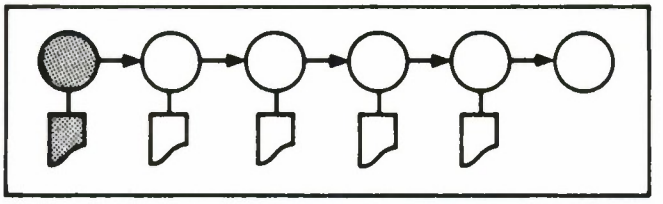
*NOTE: IMPACT IS INDICATED BY:

+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SIGN = NO PRESUMED DIRECTION

(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE II : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Data Processing Equipment						
X_{51} - First program on computer	+		+			
X_{53} - ADP components developed concurrently	+		+			
X_{68} - Number of agencies concurring in design	+		+			
Development Environment						
X_{69} - Customer inexperience	+		+			
X_{80} - Number of sources of system information	+		+			
X_{81} - Accessibility of system information	+		+			
X_{88} - Quality of resource documents	-		-			
X_{89} - Availability of special tools	-		-	38		
X_{90} - Degree of standardization in policy and procedure	-		-			



ADDITIONAL COMMENTS ON SELECTED COST FACTORS

X_{13} - Total Source Instructions Written. The rationale for using total source instructions written (or also, perhaps, X_{10}, Total Object Instructions Delivered, X_{17}, Number of Conditional Branches) as a cost factor for estimating the cost of the planning is that this variable is a principal measure of program size. The hypothesis is that larger sized programs require more thought and effort to make an estimate of their costs. Most of the time for a new ADP system, an estimate of this cost factor will be difficult to obtain.

X_{41} - Number of Subprograms. Division of a large computer program into subprograms may be a "natural" classification of data processing tasks, may help achieve a more modular design for ease of maintenance and also may facilitate the division of labor in computer programming. If separate groups of personnel are to develop these subprograms, then separate cost breakdowns should be made for these subprograms. Therefore, the number of these subprograms would substantially affect the cost of making the total estimate.

X_{84} - Number of Functions in the System. The number of functions in the system would affect the cost of making an estimate in at least two ways. First, the larger the number of functions, the more factors to be considered in attempting to determine the magnitude of the programming job. Second, the larger the number of functions affected, the greater the task of determining the present costs for the cost comparison (Figure 7).

X_{89} - Availability of Special Tools. For the cost estimation step, special tools to aid in the estimation process would include worksheets and planning factors such as those found in this Handbook. Also computer programs, based on equations and planning factors found in the Handbook, could be prepared to automatically do portions of the cost estimating task.

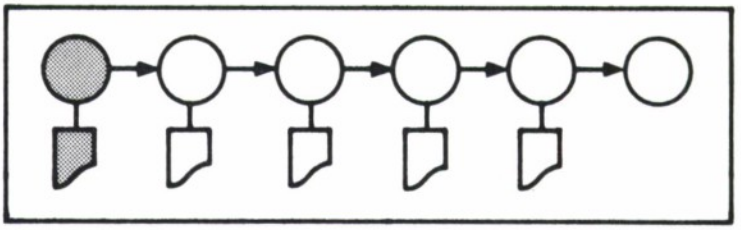


TABLE III: PLANNING FACTORS

TYPE: VARIOUS SUBJECT: ESTIMATION OF TASKS WITHIN ACTIVITY

TASK	ESTIMATION RULE
Establish System Characteristics	
Select Planning Factors	Up to one man day per subprogram.
Calculate Programming Costs	
Determine Project Costs Other than Programming	
Check Reasonableness of Estimates	Allow one man day for each expert consulted. Allow up to one man day per subprogram for each alternative estimate.
Prepare Summary Budget Plan	Allow one man day.
Determine Costs of Existing System	Costs of this step vary from those of an off-hand estimate to an elaborate study. For some concepts and methods, see (28).
Determine Economic Feasibility of Proposed System	Allow 1-2 man days for summarizing data and briefly documenting recommendations.

Costs of this activity may range from 1-5 man days for small programs (under 1000 object instructions), to 6 man months for larger (> 30,000 object instructions) efforts.

If formal cost-effectiveness studies are planned, requiring estimates and evaluation of a series of alternatives (note Figure 2), allow about 1-2 percent of the total cost of project--with a minimum of 3 man months.

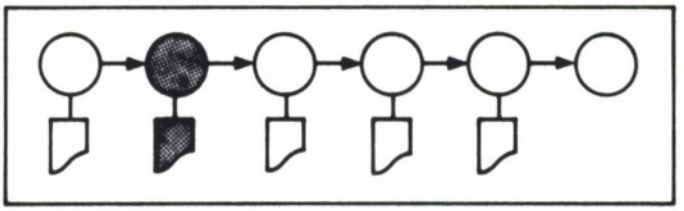


TABLE IV: ACTIVITY DEFINITION

ACTIVITY: INFORMATION PROCESSING SYSTEM ANALYSIS AND DESIGN

DESCRIPTION: The process of determining the detailed requirements for improved information processing and planning of a system, plus a set of computer programs capable of fulfilling them, is divided into two parts--System Analysis and System Design. The first part, the analysis (sometimes called problem formulation), consists of investigating the particular information processing problem that may be solved by new or improved automatic data processing methods; the second design consists of attempts to devise a satisfactory solution to the data processing requirements involved. In the broadest sense, the problem and its solution may involve the design of a far-flung network including communications displays, data equipment for sensors or missiles, computer operating procedures, and computer programs. In its narrowest sense, Analysis and Design work as part of computer programming may only include the design of a change to a computer program in an existing system.

Generally, the mission of the analyzing and synthesizing process is to devise the most effective and efficient organization of system components including computer program functions and elements possible within the constraints of available manpower, funds, and time, to perform the required information processing functions. Ideally, this selection of a solution should be made on the basis of cost/benefit comparison of feasible alternatives, so a prime use of the material in this Handbook is to help managers in making such decisions.

TABLE IV : ACTIVITY DEFINITION (CONT'D.)

(AFSCM 375 context: Information System Analysis and Design begins in the Conceptual Transition Phase, after issuance of a Specific Operational Requirement (SOR), an Advanced Development Objective (ADO), or an Operational Support Requirement (OSR), and ends during Phase A, Prepare for Contractor Definition, with the issuance of the System Specification. The design part of Analysis and Design occurs during Phase B, Contractor Definition. The resulting document, a firm definition of detailed functions, is equivalent to the "Contract End Item Detail Specification (Computer Program)--Part I.")

TASKS:

Analyze System Requirements. Determine the operational requirements of the system and evaluate their completeness, feasibility, and compatibility with other systems by studying the original definition of the problem and its references and by contacting and coordinating with user personnel.

Analyze the User's Environment. Study the user's current environment and operations, to determine how any new system will be employed, where any new operations will be located, and what the information processing responsibilities of the user are (especially in processing inputs from and outputs to other organizations); and to determine the effectiveness and deficiencies of existing data processing operations that might be improved by new or changed information processing.

Analyze Computer Program Production Requirements. Determine the requirements for computer program production and test, including the adequacy of available tools, amount and kind of programming languages, operating systems, and other programming support, the tools needed to produce any new computer programs, existing computer operations, experience with the proposed computer, and the projected availability of the computer and facility, and the availability of back-up equipment.

TABLE IV : ACTIVITY DEFINITION (CONT'D.)

Analyze Similar and Interfacing Systems. Determine if there are systems, subsystems, procedures, tools, and techniques already in production or use or planned, that may influence the proposed new system.

Prepare Design Requirements and Specifications. Develop the specifications for the total information processing system including the system configuration that is expected to meet requirements. Prepare a functional description to serve as a basis for the other development activities and to promote the user's understanding of the system. Develop the design requirements for the computer program system part of the total information processing system.

Obtain User's Concurrence. By a process of review and revision, obtain concurrence on the system specifications and the design requirements for the computer program.

PRINCIPAL INPUTS:

User's requirements including mission statements.

User's environment including system constraints and interfaces, SOPs, and description of existing system.

Descriptions of the design and operation (including experience) for subsystem and equipment components.

Production constraints and environment.

Organization charts, responsibility charts, and job descriptions.

Comparative documents (case histories, planning, and estimating documents from other similar projects, cost studies and reports).

Description of available tools (to the computer program developer) for computer program development such as languages, compilers, assemblers, utility systems, monitors, test data generation, and recording and reduction systems.

Descriptions of the computers, and other equipment, the machine language and command structure (representation of instructions and data in the computer).

TABLE IV : ACTIVITY DEFINITION (CONT'D.)

	Reports on EAM, computer, and support program usage experience and delivery dates.
	Details of computer system to be produced (real time, relocatable, task-oriented, cyclic versus stacked job, etc.).
	Computer operating procedures.
PRINCIPAL OUTPUTS:	Requirements analysis--operational and developmental.
	System design specifications including the design requirements for the computer programs and the data base.
	Plans for system test including the criteria for judging performance.
	Replanning inputs--changes to costs and schedules obtained from improved knowledge of problem and its solution.
	Report on the selection of the system design including rationale such as the results of the cost/benefit analysis on the alternative ways to accomplish the automatic data processing.

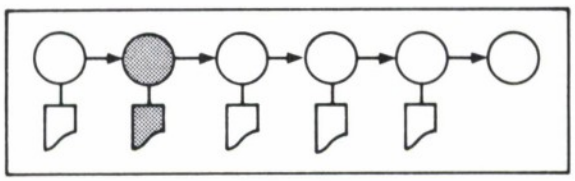


TABLE V: COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_1 - Vagueness of design requirements definition	+		+			
X_2 - Innovation required	+		+			
X_3 - Lack of knowledge of operational requirements (the problem)	-		-			
X_4 - Number of organizational users	+		+			
X_5 - Number of ADP centers	+		+			
X_7 - Response time requirements	-		-			
X_8 - Stability of design	-		-			
X_9 - On-line requirements	+		+			
X_{78} - Complexity of system interface with other systems	+		+		44	
X_{82} - Degree of system change expected during development	+		+			
X_{84} - Number of functions in the system	+		+			
X_{85} - Number of system components	+		+			

*NOTE: IMPACT IS INDICATED BY:

+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SIGN = NO PRESUMED DIRECTION

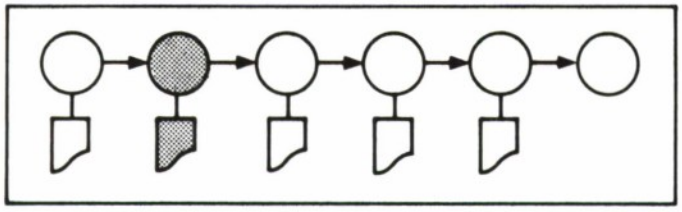
(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE V : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Program Design & Production						
X_{18} - Number of words in data base	+		+			
X_{19} - Number of classes of items in data base	+		+			
X_{20} - Number of input message types	+		+			
X_{21} - Number of output message types	+		+			
X_{22} - Number of input variables	+		+			23
X_{23} - Number of output variables	+		+			
X_{24} - Number of words in tables and constants not in data base	+		+			
X_{40} - Stringent timing	+		+			
$X_{45.1}$ - Internal documentation	+		+	48	57, 44	
X_{46} - External documentation	+		+			
$X_{47.1}$ - Total number of external document types written	+		+		57, 44	
$X_{47.3}$ - Total number of internal document types written	+		+		57, 44	
Data Processing Equipment						
X_{53} - ADP components developed concurrently	+		+			
X_{54} - Special display equipment	+		+			
Personnel						
X_{61} - Percent senior programmers	-		-			
X_{79} - Average analyst experience with similar systems (see X_{63})	-		-			
X_{64} - Percent programmer design participation limit	-		-			
X_{65} - Personnel continuity	-		-			

TABLE V : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
X_{87} - Percent senior analysts	-		-			
X_{92} - Personnel turnover	+		+			
Development Environment						
X_{67} - Lack of management procedures (2)	+		+			
X_{68} - Number of agencies concurring in design	+		+			
X_{69} - Customer in experience	-		-			
X_{75} - Number of man trips	+		+			
X_{79} - Security classification level	+		+			
X_{80} - Number of sources of system information	+		+			
X_{82} - Accessibility of system information	-		-			
X_{86} - Number of system components--not "off-the-shelf"	+		+			
X_{88} - Quality of resource documents	-		-			
X_{90} - Degree of standardization in policy procedures	+		+			
X_{94} - Number of official reviews of documents	+		+			



ADDITIONAL COMMENTS ON SELECTED COST FACTORS

X_{45} - Internal Documentation. "On the surface, documentation appears to be a time-consuming, laborious chore. But when the subject is closely examined, it becomes obvious that the bulk of the documentation is created as a result of doing a good systems job. This will be true unless the documentation requirements are so elaborate that to conform with them would be as time-consuming as doing the complete systems job." (52)

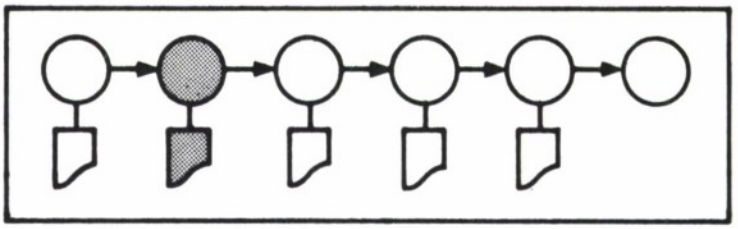


TABLE VI: PLANNING FACTORS	
TYPE: VARIOUS	SUBJECT: ESTIMATION OF TASKS WITHIN ACTIVITY
TASK	ESTIMATION RULE
Analyze System Requirements	
Analyze User's Environment	
Analyze Computer Program Production Requirements	Up to 9 man weeks (18)
Analyze Similar and Interfacing Systems	Up to 10 man weeks depending on nature of project (18)
Prepare Design Requirements and Specifications	Estimated at 10-15 percent of total project man months, exclusive of maintenance (18)
Obtain User's Concurrence	Three man days per design document per agency contacted, plus allowances in elapsed time for travel (18)

(page 50 blank)

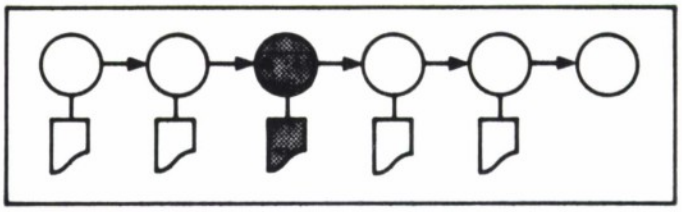


TABLE VII: ACTIVITY DEFINITION

ACTIVITY: COMPUTER PROGRAM DESIGN, CODE, AND TEST

DESCRIPTION: This activity covers all work necessary to produce the computer program end product in accordance with the detailed specification of requirements for the computer program including design, code, test (debug) and documentation work for the entire program as well as subprograms (runs, segments, individual programs).

(AFSCM 375 context: Computer program design occurs during the Acquisition Phase, and contributes to the "Contract End Item Detail Specification (Computer Program)--Part II" and is an input to the Critical Design Review (CDR). Computer program coding and test occur during the Acquisition Phase and include Category I testing and evaluation of the computer program by the developer. This step results in a complete "Contract End Item Detail Specification (Computer Program)--Part II," which is an input to the First Article Configuration Inspection (FACI). The testing in this activity includes computer program functional test that occurs in the Acquisition Phase, and is equivalent to Category I Formal Qualification testing. For systems developed according to the AFSCM 375 series, Category I Formal Qualification testing is usually conducted at the facility designated for Category II testing.)

TASKS: Develop Program System Test Plans. Develop and document program system test requirements, test plans, and test designs to provide the specific plans and criteria for the computer program system.

TABLE VII : ACTIVITY DEFINITION (CONT'D.)

Design Programs. Design and document the entire computer program system and individual programs and routines that have been identified. Determine input/output message formats.

Design Individual Program Files. Develop and define the form of the data elements to be manipulated by each program, lay out storage allocations, and document program data structures.

Establish Program System Files. Develop and maintain a central accounting system for data used by more than one program in the program system. Document the central data file structure and the procedures for maintaining it. Periodically issue listings of the central file contents.

Code the Programs. Translate flow diagrams and other statements of program designs into coded instructions.

Desk Check the Programs. Desk check program code by looking for illegal expressions, erroneous data references, program logic errors, program inefficiencies, and deviations from program specifications.

Become Familiar with the Test Environment and Procedures. Review the current procedures for using the computer, the utility system, and other support systems, using the requirements as a guide.

Compile and Check Program Code. As individual blocks or subroutine code are written in either symbolic assembly language or procedure-oriented language, assemble or compile each block into machine-readable (binary) form, check the listings for errors, correct the code and recompile. Continue this process until a satisfactory compiled program or routine is obtained.

Test Individual Programs. Within the requirements of the plans for program testing, run performance tests of the individual programs to isolate and correct errors. Rerun tests until all program requirements and design specifications are satisfactorily met.

TABLE VII : ACTIVITY DEFINITION (CONT'D.)

	Test Program Subsystems. Within the requirements of the plans for program testing, run program subsystem tests for physical integration of functionally interdependent blocks of programs to isolate and correct failures to meet program specifications. (Program systems consisting of only one individual program will not require this task.)
	Functional Test of Complete Program System. Run performance and demonstration tests of the complete program end product to isolate and correct program system malfunctions and demonstrate that the program system meets all specifications. (This test is done at the developer's facility, using a simulated total information system environment. When user's and developer's facilities are the same (e.g., in-house computer programming on the same computer to be used for operations), this task may be omitted.)
PRINCIPAL INPUTS:	System specifications with computer program requirements, including functional descriptions, system configuration, system flow charts, data element inputs and outputs, and a description of data base.
	User manuals for computer and associated software such as operating systems and compilers.
	Schedules and budgets.
PRINCIPAL OUTPUTS:	Test plans and requirements for computer programs.
	Program system test design.
	Program specifications--both system and individual program.
	Broad, detailed flow diagrams.
	Computer input and output formats.
	Coded source program statements (listing).
	Object program in binary, on cards or tape.

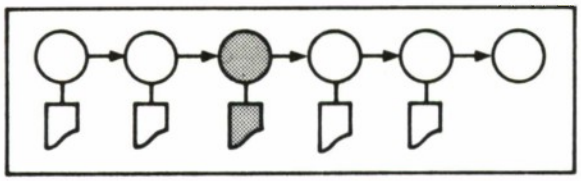


TABLE VIII : COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_1 - Vagueness of design requirements definition	+2B	+2B	+2B	86		
X_2 - Innovation required	+2B	+2B	+2B	80, 83, 86		
X_3 - Lack of knowledge of operational requirements	+2B	+2B	+2B	77, 85		
X_4 - Number of organizational users	1B	1B	+2B	80, 81		
X_5 - Number of ADP centers	+3	+2	+2	87		
X_6 - Complexity of program interface with other systems	+2B	+2	+2B	80, 82		
X_7 - Response time requirements	+4A	+4A	+4A		13, 21	
X_8 - Stability of design	+4	+3	+4	77-81, 83, 85-87		
X_9 - On-line requirements	1B	1B	+2B		13, 21, 25, 44	
X_{78} - Complexity of system interface	+	+	+		44	
X_{83} - Degree of system change expected during operations	+	+	+			
X_{84} - Number of functions in the system	+	+	+			
X_{85} - Number of system components	+	+	+			

*NOTE: IMPACT IS INDICATED BY:

CORRELATION	SIGN	OTHER CONSIDERATIONS
4 = HIGH	+ = VARIES DIRECTLY	A = HIGH CORRELATION, BUT ALSO CROSS-CORRELATION
3 = MODERATE	- = VARIES INVERSELY	
2 = LOW	NO SIGN = NO PRESUMED DIRECTION	B = STRONG INTUITIVE APPEAL
1 = INDETERMINATE		

(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE VIII: COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
					OTHER	
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	DESCRIPTIVE	ANALYTICAL
X_{86} - Number of system components not off-the-shelf	+	+	+			
Program Design and Production						
X_{10} - Total object instructions delivered	+4	+4	+4	60	13	23
X_{11} - Percent delivered object instructions reused	-	-	-			
X_{12} - Total nondelivered object instructions produced	+	+	+			
X_{13} - Total source instructions written	+4	+4	+4			23
X_{14} - Percent source instructions written in POL	-	-	-			
X_{15} - Percent of total object instructions discarded	+	+	+			
X_{16} - Percent of total source instructions discarded	+	+	+			
X_{17} - Number of conditional branches	+2	+4	+2	85		23
X_{18} - Number of words in data base	1B	1B	1B	80		
X_{19} - Number of classes of items in data base	1B	1B	1B	79,81,83, 86	13	23
X_{20} - Number of input message types	1B	1B	1B			23
X_{21} - Number of output message types	1B	1B	1B			23
X_{22} - Number of input variables	1B	1B	1B			23
X_{23} - Number of output variables	1B	1B	1B			
X_{24} - Number of words in tables and constants not in data base	+	+	+			
X_{25} - Percent clerical	-2	+2	-2	82		
X_{26} - Percent mathematical	+2	-2	2	77		
X_{27} - Percent I/O	-2	-2	-2			

TABLE VIII : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
					OTHER	
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	DESCRIPTIVE	ANALYTICAL
X_{28} - Percent logical control	+2	+2	+2			
X_{29} - Percent self-checking	-2	-2	-2			
X_{30} - Percent information storage and retrieval	+2B	+2	+2	77,82,86		
X_{31} - Percent data acquisition and display	-2	+2	+2			
X_{32} - Percent control or regulation	+2	+2	+2			
X_{33} - Percent decision making	+2	+2	+2			
X_{34} - Percent transformation	-2	-2	-2			
X_{35} - Percent generation	+2	+2	+2	78,85,87		
X_{36} - Average operate time	1B	1	1B			
X_{37} - Frequency of operation	+4	+4	+4	78,79, 81-83		
X_{38} - Insufficient memory	+4A	+4A	+4A			
X_{39} - Insufficient I/O capacity	+	+	+			
X_{40} - Stringent timing requirements	+4	+4	+4	79,81,83, 84,87		
X_{41} - Number of subprograms	+4	+4	+4	77,79,80, 81,84,87		
X_{42} - Programming language	+4	+4	+4A	60,77,84, 85		
X_{43} - POL expansion ratio	-2	-2B	-2	62		
X_{44} - Support program availability	B	B	B		58	
$X_{45.1}$ - Internal documentation, written	+4A	+2	+3A		44, 57	
$X_{45.2}$ - Internal documentation, available	-	-	-			
X_{46} - External documentation	+4	+4	+4A	78,82,84, 87		
$X_{47.1}$ - Total number of external document types, written	+2B	+2	+2B	78,79,84	44, 57	
$X_{47.2}$ - Total number of internal document types, available	-	-	-			

TABLE VIII : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

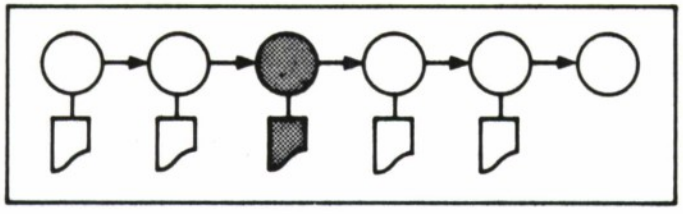
FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
$X_{47.3}$ - Total number of internal document types, written	+1B	+1B	+1B	78,79		
X_{48} - Type of program						
$X_{48.1}$ - Business	-4	-4	-4	77,82,83, 86,87		
$X_{48.2}$ - Scientific	+2B	-2B	-2B			
$X_{48.3}$ - Utility	+4	+4	+4	78,79,80, 83	10, 50, 60	
$X_{48.4}$ - Other	+2B	+2B	+2B			
$X_{48.5}$ - Stand-alone	1B	+2B	+3B	77, 80	13	
X_{89} - Availability of special tools	-	-	-	64	58	
X_{93} - Output volume	+	+	+			23
X_{94} - Input volume	+	+	+			23
Data Processing Equipment						
X_{49} - Compiler or assembler used	4	3	1	62		14
X_{50} - Developmental computer used	2B	2B	2B			
X_{51} - First program on computer	+4	+4	+4	77,79,81, 82,83,87		
X_{52} - Average turnaround time	2B	2B	2B	78,85		
X_{53} - ADP components developed concurrently	+4	+2B	+4	62,77,80, 87	33	
X_{54} - Special display equipment	+4	+4	+2	80,81,82, 85,87		
X_{55} - Core capacity	+2B	-2	+2B	84		23
X_{56} - Random access device used	+4	+4	+2B	77,78,80, 81,82,85, 87		
X_{57} - Number of bits in word	+4	+3	+4	78,84,85, 87		
X_{58} - Memory access time	-2	1B	-3A			
X_{59} - Machine add time	-3A	1B	-3A			
X_{60} - Computer cost	-4	-3	-4			23

TABLE VIII : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Personnel						
X_{61} - Percent senior programmers	-4A	-3A	-3	86		
X_{62} - Average programmer experience with language	-3	-1B	-2B	63, 85		
X_{63} - Average programmer experience with application	-2	1B	-2	84	63	23
X_{64} - Percent programmer design participation	1B	-4	-3	78,80,82, 84, 87		
X_{65} - Personnel continuity	-4	-4	-4	79,82,83, 85-87		
X_{66} - Maximum number of programmers	+4A	+4A	+4A	63	38	
X_{87} - Percent senior analysts	-	-	-			
X_{92} - Personnel turnover	+	+	+			
Development Environment						
X_{67} - Lack of management procedures	-2	-2	+2			
X_{68} - Number of agencies concurring in design	+2	+2	+2			
X_{69} - Customer inexperience	1B	1B	1B			
X_{70} - Computer operated by agency other than developer	+2	+2	+2		60	
X_{71} - Different site for programming and operation	1B	1B	1B	80		
X_{72} - Different computers for programming and operation	+4	+4	+4	63,77,78, 79,84		
X_{73} - Closed or open shop facility operation	-3A	-4A	-3A	63	8, 48	
X_{74} - Number of locations for program development	+4	+3A	+4A	82		
X_{75} - Number of man trips	+4	+4	+4	77,79,80, 83,86,87		
X_{76} - Developed by military organization	1B	-4	-4	77,87,79, 80,81,83		

TABLE VIII : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
X_{77} - Developed on time-shared computer	-2B	-2B	-2B			34
X_{79} - Security classification	+	+	+			
X_{88} - Quality of resource documents	-	-	-		52	
X_{90} - Degree of standardization in policy & procedure	+	+	+			
X_{91} - Number of official reviews of documents	+	+	+			



ADDITIONAL COMMENTS ON SELECTED COST FACTORS

X_{10} - Total Object Instructions Delivered. Obviously, larger programs require larger amounts of resources. There is some statistical evidence (32) that larger programs also involve a disproportionate increase in man months; that is, the man months required per thousand delivered instructions increase as the total number of instructions in the program increases. That a program with many instructions costs more per instruction than one with fewer instructions is a commonly held belief (13, 36). One reason that this may be so is that larger programs must usually be broken down into smaller pieces for individuals to work in; this complicates both the program design problem and the subsequent assembly and checkout of the various subprograms (27). The evidence on this point is, however, not conclusive.

X_{42} - Programming Language. One authority states the following on the general question of procedure-oriented language versus assembly language programming (55):

"a) In general, there is no appreciable difference in the amount of training needed to acquire professional competence in the use of either a procedure language or an assembly language.

"b) The use of an appropriate procedure language, by reducing the number of steps in the source program and by easing the job of program modification, can significantly reduce the amount of effort needed for program production and maintenance.

"c) The use of a procedure language instead of an assembly language improves the communication of algorithms between programmers--primarily by reducing the number of steps needed for their expression. This improvement results in a reduced need for detailed, step-by-step program documentation.

"d) Procedure languages, because they are largely machine independent and because they make programs easier for programmers to read and to modify, can greatly facilitate the transfer of programs between different computer types. Of course, the transfer of programs that are system-dependent is not always practical, however machine-dependent, readable, and changeable they are.

"e) Object-code efficiency is often not an important factor; nevertheless, with a good compiler, an average programmer will usually turn out code that is just about as efficient as the code he would produce using an assembler.

"f) The use of a procedure language instead of an assembly language does not increase the difficulty of obtaining program timing estimates, since these estimates seldom involve the execution times of individual program steps."

A definite trend from machine-oriented languages to problem-oriented languages is frequently cited in the literature (7).

The resource cost rates cited in this Handbook provide, for the specific sample of programs studied, some evidence of the relative overall expenditure of resources for the total program design, code, and test effort for several procedure-oriented languages. For source-program compilation time only, a test was made (IBM 7090 with IBSYS) with four programs written in both FORTRAN and COBOL (7). The following results of this test indicate longer compile times for COBOL:

Type of Problem	Number of Cards in Source Program		Compilation Times (in minutes)		% Increase in Compilation of COBOL over
	FORTRAN	COBOL	FORTRAN	COBOL	FORTRAN
Job Order Cost Calculation	18	72	0.5007	0.7989	59.6
Sort Routine	25	56	0.4890	0.8108	65.8
Hourly Payroll Computation	35	115	0.5099	0.8390	64.5
Order Processing Cycle	22	288*	0.5040	1.1111*	120.46*

*Includes additional output report on rejected orders.

X_{43} - POL Expansion Ratio. Working with the better JOVIAL compilers, the size of the object program generated from JOVIAL source statements may be 10-15 percent larger than if the source program had been coded in machine languages (62). Likewise, in terms of size of the object program, COBOL programs average 10-15 percent larger than their machine language counterparts (37). This "excess generation" would be a part of the POL expansion ratio, since this ratio is computed by dividing the total number of object instructions by the total number of source language statements.

X_{49} - Compiler or Assembler Used. The impact of the compiler-hardware combination on compilation cost is illustrated by the following (14):

Computer	Typical Compilation Speed, COBOL Statements per minute	Prime Shift Cost per minute ($)	Cost per 100 COBOL Statements ($)
IBM-7010	550	1.88	.34
H-1800 III	1000	2.31	.23
B-5000	450	2.15	.48
U-1107	700	3.50	.50
IBM-7074	30	2.66	8.87
IBM 7080	29	5.02	17.31
H-400	22	.78	3.55
RCA-301	12	.54	4.50
IBM-1401	10	.52	5.20

X_{53} - ADP Components Developed Concurrently. Experience by the USAF in the development and programming of special-purpose computers such as AN/FSQ-7, AN/FSQ-8, AN/FSQ-31, AN/FSQ-11, has revealed that the checkout of new unproven hardware from one company and the unproven software (see Information System Integration Test Activity, Section VI) to make it perform from another can add significantly to the expense of the project. A major problem is the difficulty in establishing responsibility when either hardware or software fails to perform to specifications (24).

X_{62} - Average Programmer Experience with Language. "A knowledge of the 'compiler's ways' is invaluable to the user and can account for differences of as much as 40% in running time, and even more in object program size." (37)

X_{66} - Maximum Number of Programmers. Absolute values of the resource units, e.g., number of computer hours, correlate directly with the maximum number of programmers assigned (X_{66}), primarily because more programmers must be assigned to the larger development efforts in order to attempt to meet the imposed schedules. It should not be inferred from this, however, that larger programming staffs are inherently less efficient, e.g., have lower production rates, than smaller groups. If the staff is properly managed, there are no a priori reasons why a large staff should not operate as efficiently as its smaller counterpart (38). For a contradicting view on this point, see (58).

X_{72} - Different Computers for Programming and Operation. An advantage claimed for problem-oriented languages is that a program written for one computer can be readily recompiled to run on another. But there are still costs of making this conversion. "Tests show that converting a COBOL source program from one computer to another involves only 10 to 40% of the time required to write the original program in COBOL. It should be stressed that this time is a percentage of actual 'coding' time, and not the total time required to analyze, design and code. If the two machines are very similar in design and have the same collating sequence, the original coding time might be reduced by 95%." (37)

X_{73} - Closed or Open Shop Facility Operation. Arguments can be made for either type of facility operation (8). The following advantages are claimed for the "open shop":

1. A programmer can discover more errors per test shot.

2. By observing the program run, the programmer may be able to detect operational weaknesses in the program.

3. Programmers may be better machine operators, and be more careful with their own programs, resulting in fewer operator errors during testing.

4. Testing time is reduced because of quick turnaround.

On the other hand, the following arguments are advanced in favor of the "closed shop":

1. The programmer may foul up his program by experimenting with the console after a hang-up.

2. Programmers waste time waiting for machine availability, and may be less efficient than a regular operator getting on and off the machine.

3. Programmers tend to waste machine time.

4. If test instructions are prepared for operators, the programmer tends to organize test shots better.

The research of this project is not conclusive, but computer hours per 1000 object instructions tend to be larger in open-shop operations; correlation coefficient of computer hours per 1000 object instructions with variable X_{73} (coded: open shop = 0, closed shop = 1) is -.21 for the total sample of 169 data points. From this statistic, one infers that computer usage rates are lower in closed shops.

In a System Development Corporation survey of 30 organizations (34), the following describes the tendency to favor the closed-shop system:

	Scientific Programming (%)	Business Programming (%)	Both (%)
Open Shop	39	12	27
Closed Shop	61	88	73

X_{77} - Time-Sharing. In the first known study of the relative performance of programmers working under conditions of on-line (time-sharing) and off-line access to computers, the following conclusions were suggested (34):

1. On-line operations are superior to off-line in terms of man hours spent debugging.

2. On-line operations tend to use more computer time than off-line programming.

3. There are striking individual differences in programmer performance.

X_{89} - The Availability of Special Tools. It has been claimed that experience at General Electric shows that the actual time spent in problem definition and coding can be reduced with the use of decision tables. "Users report up to 90% lower programming costs because detailed flow charting and coding are virtually eliminated. Other users report 50% lower application costs because of this, plus improved systems design and check-out efficiency." (52)

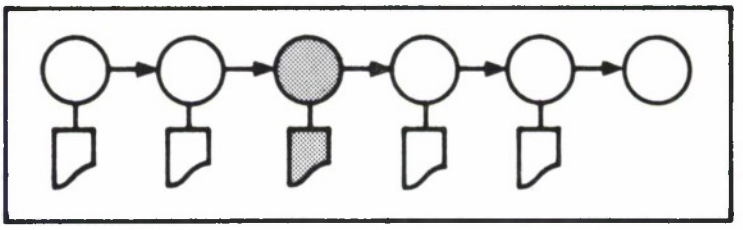


TABLE IX: PLANNING FACTORS

TYPE: VARIOUS SUBJECT: QUOTES WITHOUT COMMENT

1. "A programmer can prepare, check out, debug, and document about 1 instruction per hour." (36)

2. "In the past, preparation costs of computer programs averaged $8 per instruction...new techniques have cut this cost to about $.80 per instruction." (43)

3. "For large scale programs, approximately 50 instructions can be developed per hour of computer time." (36)

4. "The (compilers) in existence today have taken on the order of 15 to 25 man years of experienced programming talent, spread over about three calendar years." (59)

5. "The charges for direct salaries and supplies approximately equal to first shift rental of the computer in an average installation." (45)

6. "Less than 5% of the total cost of a computer center is in coding." (12)

7. "For real-time systems, of the time between the appearance of the first unit function-statements and the beginning of program acceptance tests, at least 2/3 should be allotted to build-up and checkout of successive 'packages' culminating in the completed program; and an 'executive' routine, including control of test inputs and recording, should be ready, together with the first nucleus of program units, no later than the end of the first third of this interval." (25)

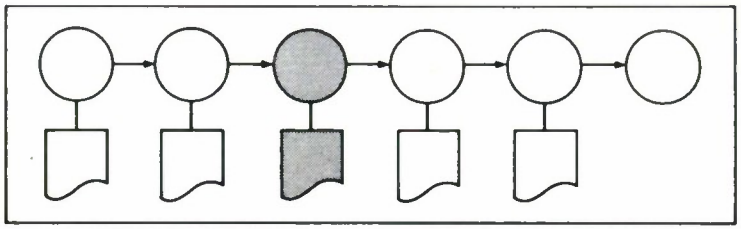


TABLE X: PLANNING FACTORS

TYPE: UNIT COST

SUBJECT: RESOURCE RATE PER 1000 SOURCE INSTRUCTIONS

TYPE OF COMPUTER PROGRAM	NO. OF DATA POINTS	RESOURCE COST RATE														
		MAN MONTHS $\frac{Y_1}{X_{13}} \times 10^3$					COMPUTER HOURS $\frac{Y_2}{X_{13}} \times 10^3$					ELAPSED TIME (MO.) $\frac{Y_3}{X_{13}} \times 10^3$				
		MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN
MOL	123	100.00	.15	10.30	4.00	6.34	331.25	.23	45.78	15.00	31.39	40.00	.07	7.04	1.47	4.13
POL	46	46.25	.27	10.36	3.57	8.20	211.59	1.18	42.97	15.59	31.96	76.92	.16	14.46	4.79	9.11
		N.S.					N.S.					P < .01				
FORTRAN	8	38.46	1.33	14.59	4.09	12.75	211.54	1.24	71.75	11.97	43.75	76.92	2.00	26.74	5.24	18.38
JOVIAL	15	46.25	2.13	12.01	6.15	10.27	136.95	2.67	41.94	33.11	47.60	13.25	1.20	3.44	4.96	5.13
COBOL	12	28.00	.27	7.53	2.68	4.83	115.00	1.18	31.68	7.17	18.08	22.00	.23	6.01	2.57	4.42
OTHER POL	11	16.67	.57	5.10	3.23	5.73	42.68	2.77	13.77	12.90	17.18	55.56	.16	15.78	8.57	12.91
		N.S.					N.S.					N.S.				
BUSINESS	79	46.25	.15	8.27	2.73	5.75	115.00	.23	27.34	8.11	20.59	40.00	.07	7.77	2.83	5.51
SCIENTIFIC	27	38.46	.57	7.96	4.44	6.85	211.54	.25	50.69	10.44	30.67	76.92	.24	17.60	2.29	8.33
UTILITY	28	100.00	.50	18.80	5.75	10.43	331.25	2.53	74.34	38.19	62.71	21.00	.28	4.90	1.04	2.68
OTHER	35	24.50	1.49	5.11	4.63	6.46	129.03	3.86	28.58	25.00	32.00	40.00	.24	8.47	2.22	5.49
		P < .25					P < .01					P < .25				
LARGE COMPUTER	105	100.00	.19	11.99	4.75	7.94	331.25	.23	50.84	18.72	36.14	76.92	.23	11.20	1.47	5.66
MEDIUM COMPUTER	53	28.00	.15	5.15	3.05	4.62	177.77	1.25	33.29	8.70	22.34	34.84	.07	7.58	2.40	5.25
SMALL COMPUTER	11	38.82	1.23	10.75	4.00	7.11	80.00	1.11	23.73	34.69	32.05	14.93	.64	4.79	2.98	5.01
		P < .25					P < .25					N.S.				
TOTAL SAMPLE	169	100.00	.15	10.31	4.00	6.85	331.25	.23	44.91	15.00	31.54	76.92	.07	9.84	2.05	5.49

NOTES:

1) P = Probability of erroneously concluding that the population means are different (e.g., $p < .01$ indicates that if one concludes that the population means are different, the probability of error is less than 1 out of 100). If $p > .25$, it is denoted by N.S. (not significant) in this table.

2) All of the planning factors above are based on raw data that has not been winsorized (i.e., extreme values reduced) to prevent distortion by large values.

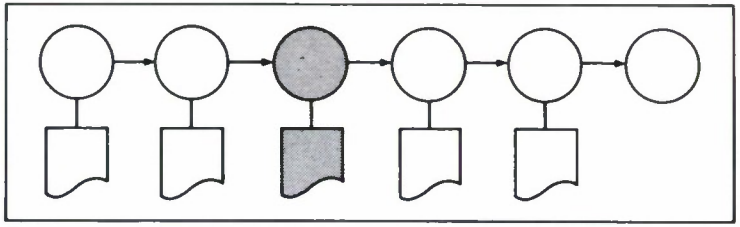


TABLE XI: PLANNING FACTORS

TYPE: UNIT COST

SUBJECT: RESOURCE RATE PER 1000 OBJECT INSTRUCTIONS

TYPE OF COMPUTER PROGRAM	NO. OF DATA POINTS	RESOURCE COST RATE														
		MAN MONTHS $\frac{Y_1}{X_{10}} \times 10^3$					COMPUTER HOURS $\frac{Y_2}{X_{10}} \times 10^3$					ELAPSED TIME (MO.) $\frac{Y_3}{X_{10}} \times 10^3$				
		MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN
MOL	123	100.00	.14	10.18	4.00	5.89	294.04	.05	42.75	15.00	29.52	40.00	.06	5.81	1.33	3.55
POL	46	9.49	.07	2.61	1.16	2.13	52.50	.30	13.74	2.86	9.76	18.43	.06	3.71	.92	2.30
		$P < .05$					$P < .01$					N.S.				
FORTRAN	8	9.49	.11	3.88	.97	2.75	50.69	.31	18.02	2.68	10.25	18.43	.17	6.95	1.14	4.50
JOVIAL	15	7.60	.66	2.31	2.50	3.07	52.50	.77	15.06	16.67	17.73	8.39	.47	1.88	1.18	1.87
COBOL	12	9.33	.07	2.53	.49	1.25	38.33	.30	10.70	1.80	5.25	7.33	.06	1.98	.41	1.08
OTHER POL	11	4.10	.12	1.57	.57	1.36	14.49	.31	3.86	2.43	3.45	11.27	.08	3.56	.74	2.64
		N.S.					$P < .05$					N.S.				
BUSINESS	79	38.82	.07	5.11	1.54	3.13	80.00	.23	17.93	3.09	11.95	18.94	.06	4.05	1.11	2.87
SCIENTIFIC	27	12.00	.14	2.90	3.14	3.55	140.00	.05	28.52	4.83	17.78	18.43	.11	4.28	1.18	2.93
UTILITY	28	100.00	.49	18.83	5.28	9.79	294.04	1.06	68.30	38.18	57.36	21.00	.10	4.03	1.03	2.03
OTHER	35	17.65	.66	4.16	4.20	5.89	129.03	3.86	28.42	20.98	30.00	40.00	.24	8.41	2.00	5.14
		$P < .01$					$P < .01$					$P < .25$				
LARGE COMPUTER	105	100.00	.07	10.51	3.17	5.50	294.04	.05	42.47	12.50	26.72	40.00	.06	5.99	1.00	3.08
MEDIMUM COMPUTER	53	16.67	.12	3.09	2.54	3.19	177.77	.31	30.22	5.20	17.57	18.94	.06	3.92	2.00	3.19
SMALL COMPUTER	11	38.82	.93	10.81	4.00	6.97	80.00	1.09	24.51	34.69	31.14	14.93	.64	4.78	2.98	4.66
		N.S.					N.S.					N.S.				
TOTAL SAMPLE	169	100.00	.07	8.94	2.93	4.87	294.04	.05	38.16	10.44	24.14	40.00	.06	5.34	1.22	3.21

NOTES:

1) P = Probability of erroneously concluding that the population means are different (e.g., $p < .01$ indicates that if one concludes that the population means are different, the probability of error is less than 1 out of 100). If $p > .25$, it is denoted by N.S. (not significant) in this table.

2) All of the planning factors above are based on raw data that has not been winsorized (i.e., extreme values reduced) to prevent distortion by large values.

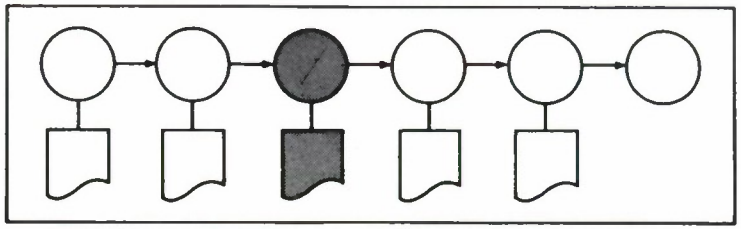


TABLE XII: PLANNING FACTORS

TYPE: UNIT COST

SUBJECT: TOTAL PAGES OF DOCUMENTATION PER 1000 OBJECT INSTRUCTIONS

TYPE OF COMPUTER PROGRAM	NO. OF DATA POINTS	DOCUMENTATION RATE														
		TOTAL $\frac{X_{45.1} + X_{46}}{X_{10}} \times 10^3$					EXTERNAL $\frac{X_{46}}{X_{10}} \times 10^3$					INTERNAL $\frac{X_{45.1}}{X_{10}} \times 10^3$				
		MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN	MAX	MIN	σ	MEDIAN	MEAN
MOL	123	1447	0	222	40	112	1067	0	120	16	43	1340	0	165	25	68
POL	46	6335	0	1175	17	315	116	0	22	3	12	6329	0	1176	8	302
		$P < .10$					$P < .10$					$P < .05$				
FORTRAN	8	6335	4	2532	14	1646	12	0	4	4	5	6329	0	2531	7	1641
JOVIAL	15	212	0	67	16	50	116	0	34	9	26	160	0	42	3	18
COBOL	12	70	0	20	17	22	20	0	8	3	7	66	0	19	10	16
OTHER POL	11	171	0	51	9	28	29	0	9	0	3	143	0	43	8	26
		$P < .01$					$P < .05$					$P < .01$				
BUSINESS	79	6335	0	754	24	176	586	0	88	3	27	6329	0	746	15	150
SCIENTIFIC	27	4710	0	911	27	252	63	0	16	3	12	4700	0	906	25	237
UTILITY	28	417	3	92	33	69	161	0	44	20	37	333	0	68	4	31
OTHER	35	1447	2	293	56	160	1067	0	178	27	66	1340	0	232	31	90
		N.S.					N.S.					N.S.				
LARGE COMPUTER	105	6335	0	802	32	213	1067	0	107	10	32	6329	0	796	11	178
MEDIMUM COMPUTER	53	1171	0	181	36	85	586	0	84	4	31	586	0	117	25	53
SMALL COMPUTER	11	520	7	179	28	126	512	0	152	9	65	282	0	97	20	61
		N.S.					N.S.					N.S.				
TOTAL SAMPLE	169	6335	0	643	33	167	1067	0	104	9	34	6329	0	633	16	132

NOTES:

1) P = Probability of erroneously concluding that the population means are different (e.g., $p < .01$ indicates that if one concludes that the population means are different, the probability of error is less than 1 out of 100). If $p > .25$, it is denoted by N.S. (not significant) in this table.

2) All of the planning factors above are based on raw data that has not been winsorized (i.e., extreme values reduced) to prevent distortion by large values.

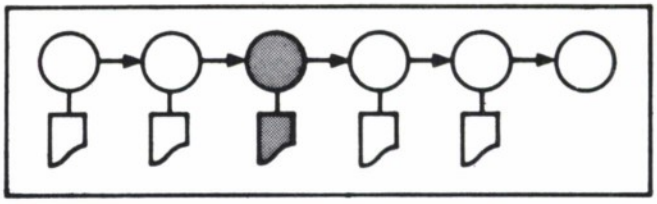
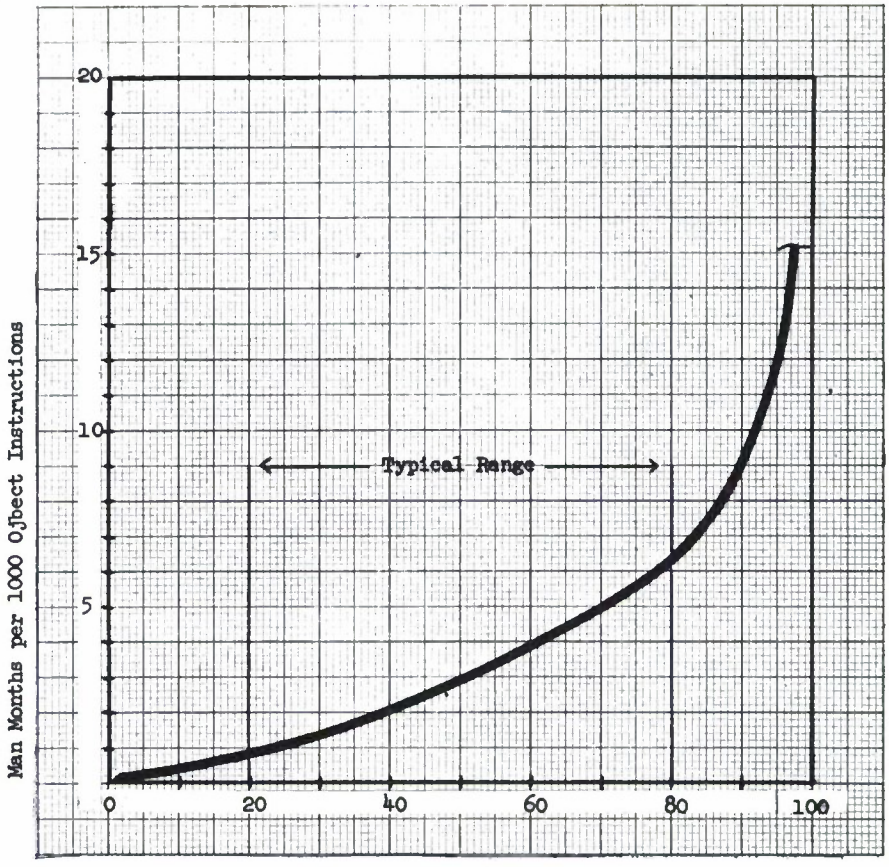


TABLE XIII: PLANNING FACTORS

TYPE: Unit Cost

SUBJECT: Man Months Expenditure Rates



Note: This curve is a plot of the man month expenditure rate per 1000 object instructions (ordinate) against the percent of the total sample of completed computer programs (absissa) that experienced this rate or less. The typical range is defined to exclude the upper and lower 20 percentiles. Example: if the estimator believed his program is more "difficult" than the median (50% on the absissa) but not atypical; i.e., as "difficult" as the extreme values, he might choose to use rates for the 60-80 percent range; then his expected expenditure rate taken from the ordinate would be 3.9-6.3 man months per 1000 object instructions.

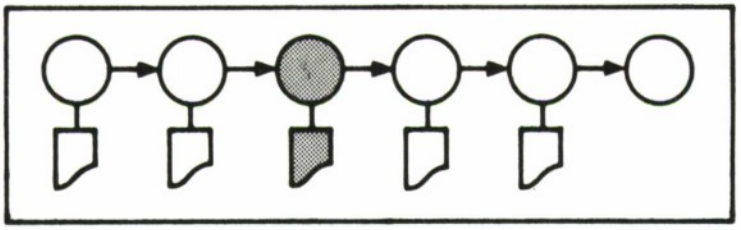


TABLE XIV : PLANNING FACTORS

TYPE: UNIT COST SUBJECT: ESTIMATION OF TASKS WITHIN ACTIVITY

TASK	RULE FOR ESTIMATING MAN MONTHS
Develop Program System Test Plans	1 man month per 10,000 estimated machine instructions (18).
Design Programs	1/2 to 1 man month per 1000 machine language instructions. 1 man month per 1000 machine language instructions for programs > 30,000 instructions (18).
Design Program Files	1 man month per 10,000 items (18)
Establish System Files	1 man month per 10,000 machine language instructions. 2 man months per 10,000 machine language instructions for programs > 30,000 instructions (18).
Code Programs	
Desk Check Programs	
Learn Test Environment and Procedures	1 man week per programmer
Compile and Check Program Code	
Test Individual Programs	See page 71.
Test Program Subsystems	See page 71.
Functional Test of Complete Program System	See page 71.

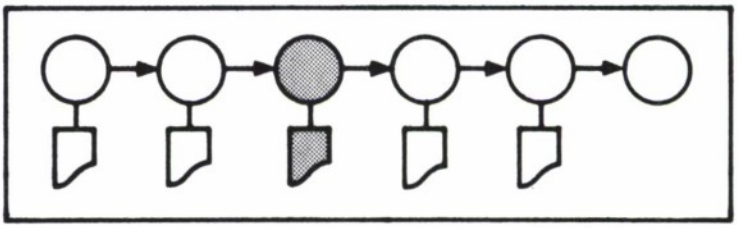


TABLE XV: PLANNING FACTORS

TYPE: PERCENT OF OTHER ITEM

SUBJECT: ESTIMATION OF TASKS WITHIN ACTIVITY

TASK	RULE FOR ESTIMATING MAN MONTHS
Develop Program System Test Plans	See page 70.
Design Programs	See page 70.
Design Program Files	See page 70.
Establish System Files	See page 70.
Code Programs	
Desk Check Programs	
Learn Test Environment and Procedures	See page 70.
Compile and Check Program Code	
Test Individual Programs	All testing requires 40-50 percent of total development effort (18). Program test requires about 20 percent of testing effort. Expect one error per 30 instructions (18).
Test Program Subsystems	All testing requires 40-50 percent of total development effort (18). Subsystem testing requires 0-30 percent of testing effort, depending on number of subsystems (18).
Functional Test of Complete Program System	About 50 percent of total testing effort, hence, about 20-25 percent of total development effort (18).

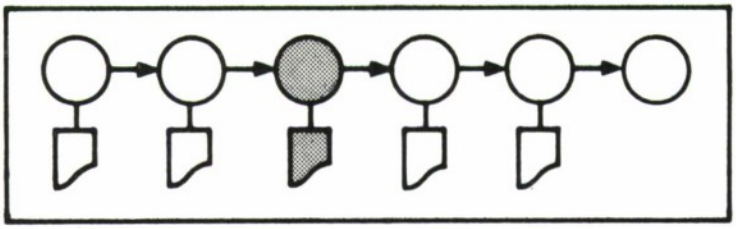


TABLE XVI: PLANNING FACTORS

TYPE: PERCENT OF OTHER ITEM SUBJECT: PROGRAM DEVELOPMENT COMPUTER TIME

	Percentage of Work Processed by Computer in 27 Firms in Southern California*		
	Scientific	Business	Both
Computer time spent in developing or modifying programs used for a specific purpose in future computer applications	46	30	40
Production operation of checked-out programs for specific tasks, e.g., payroll, data analysis	47	66	55
Computer time spent in developing programs for general applications or system control	7	4	5
TOTAL	100%	100%	100%

*Source: (34)

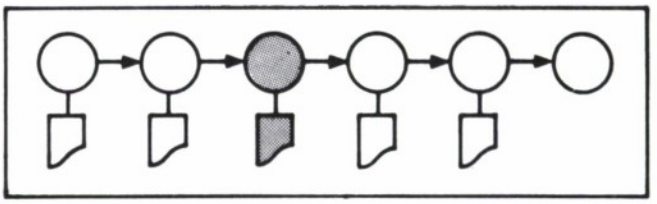
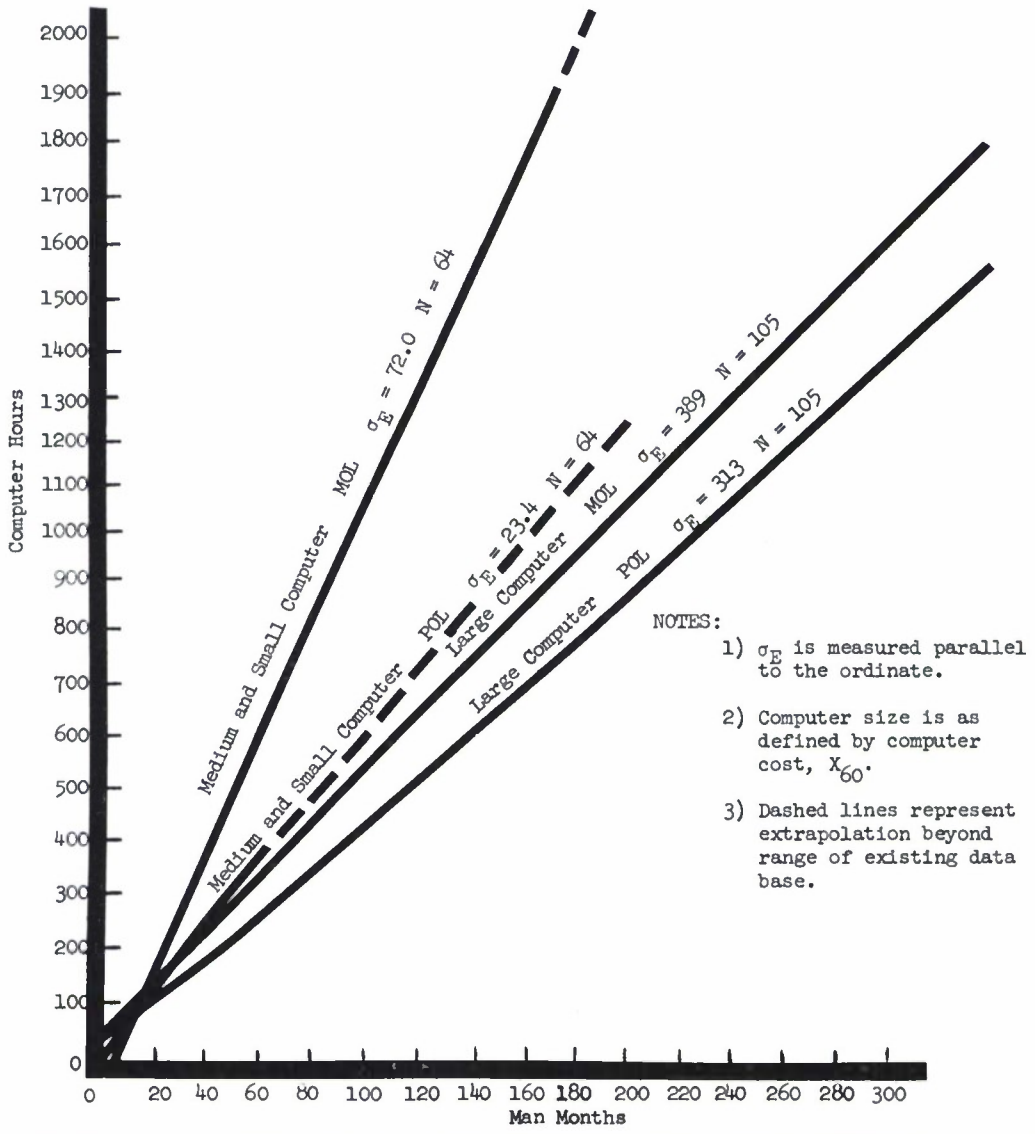
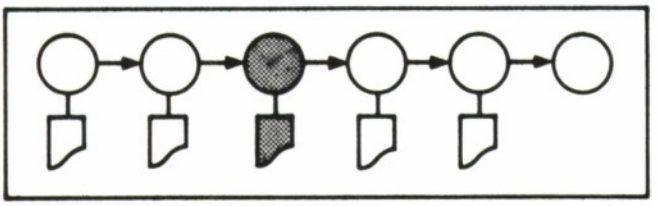


TABLE XVII: PLANNING FACTORS

TYPE: PERCENT OF OTHER ITEM SUBJECT: ESTIMATING COMPUTER HOURS FROM MAN MONTHS





COMPUTER PROGRAMMING COST-ESTIMATING EQUATIONS

The following pages contain several cost-estimating equations for computer program design, code, and test. These equations were derived by statistical analysis of the total sample of 169 data points, as well as of selected subsamples from the total sample; they represent the best predictors of the cost of the programs in this data base that the Programming Management Project was able to produce with the available resources.

In all, three subsamples were investigated: programming language (X_{42}), type of application (X_{48}), and computer size based on cost (X_{60}). These subsamples were further broken down into: MOL and POL; business programs (predominantly file-oriented, $X_{48.1}$); scientific programs (predominantly process-oriented, $X_{48.2}$); software, i.e., computer utility programs ($X_{48.3}$); other programs that could not be classified as business, scientific, nor utility, such as command and control ($X_{48.4}$); independent, or stand-alone programs ($X_{48.5}$), and programs prepared using a large-, medium-, and small-scale computer, as defined by an arbitrary cost range (X_{60}). For each of these ten subsamples, attempts were made to develop prediction equations for each of the three basic resources: man months, computer hours, and months elapsed. Equations for the total sample ($N = 169$) were developed first; then attempts were made to produce equations for subsamples that represented an improvement over the total sample. The criteria for improvement included increased (over earlier equations in references (19, 31, and 62)) intuitive appeal as well as increased statistical precision over the equations in the total sample. The measures of statistical precision include low standard error of estimate (σ_E), especially when related to the mean, and the standard deviation (σ) of the resource (i.e., man months); high correlation coefficient (r) and coefficient of determination (r^2). For those subsample equations that showed some improvement and were included in this Handbook, Table XVIII summarizes their statistical characteristics, e.g., sample size, mean, and standard error of estimate. (These statistics are defined in Glossary B.)

The development and improvement of cost estimating relationships is an iterative procedure (20, 62). A large number of combinations of subsamples and variables are possible, and many analytical trials may be necessary before the more useful and meaningful results emerge. Additional experimentation with the existing 169-point data base could include various other attempts at subsample definition, such as by size of program (31), by make of computer hardware, or, for that matter, by subsamples based on most of the variables in Glossary A; another

promising project would be an attempt to produce predictors of computer programming expenditure rates such as man months per 1000 source statements as characterized in the Planning Factors for this section.

Before using the following equations, the reader's attention is called to the coding of all variables in Glossary A. In addition, to help the reader identify the numbered variables in the equations quickly, the names of these variables have been listed with their numbers in a foldout immediately following the last equation in this Section. The selection by the Handbook user of a particular equation will depend on the appropriateness of the equation to his problem, and his knowledge of the values of the independent variables.

TABLE XVIII

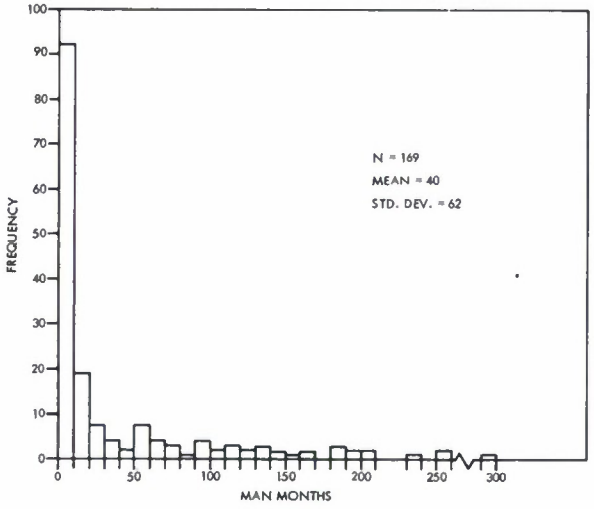
COMPARISON OF ESTIMATING EQUATION CHARACTERISTICS

Sample Name	N	Resource Estimated	Characteristics of Equation				
			Mean	σ	σ_E	r^2	r
Total Sample	169	Man Months	39.5	62.1	42.3	.58	.76
Total Sample	169	Computer Hours	237	464	320	.56	.75
Total Sample	169	Months Elapsed	9.2	7.3	4.8	.60	.77
Medium Computer	53	Man Months	15.9	32.0	10.4	.90	.95
Medium Computer	53	Months Elapsed	7.0	5.4	2.9	.71	.84
Large Computer	105	Man Months	54.2	71.5	21.6	.68	.83
Large Computer	105	Months Elapsed	10.6	8.1	5.2	.64	.80
Other Subsample	35	Computer Hours	263	395	119	.90	.95
Utility Subsample	28	Computer Hours	766	730	229	.90	.95
POL Subsample	46	Months Elapsed	8.2	8.3	6.8	.88	.94
MOL Subsample	123	Man Months	48	68	37.8	.69	.83

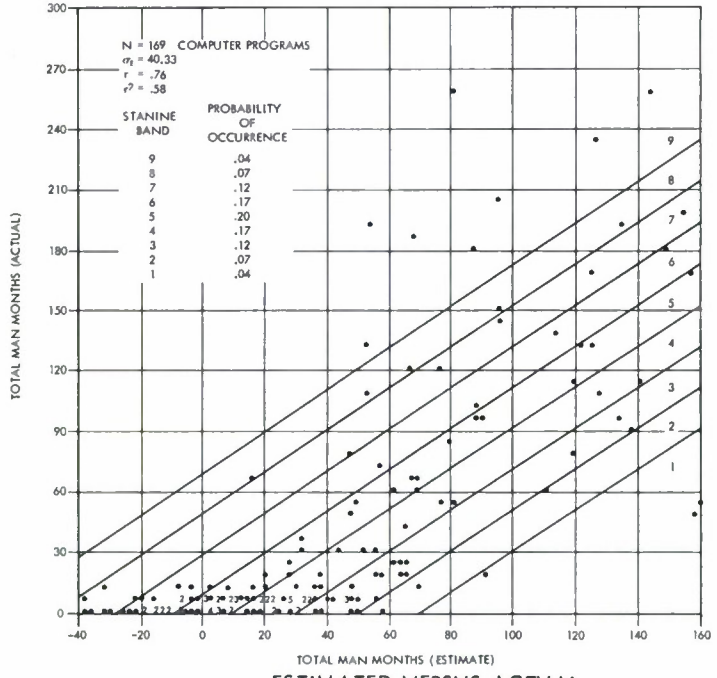
TABLE XIX :	DATA BASE: TOTAL SAMPLE N = 169	RESOURCE: MAN-MONTHS

TOTAL MAN-MONTHS = Y_1 , WHERE:

$$Y_1 = -33.63 + 9.15X_3 + 10.73X_8 + .51X_{26} + .46X_{30} + .40X_{41} + 7.28X_{42} - 21.45X_{48.1} + 13.53X_{48.5} + 12.35X_{51} + 58.82X_{53} + 30.61X_{56} + 29.55X_{72} + .54X_{75} - 25.20X_{76}$$



HISTOGRAM OF MAN MONTHS

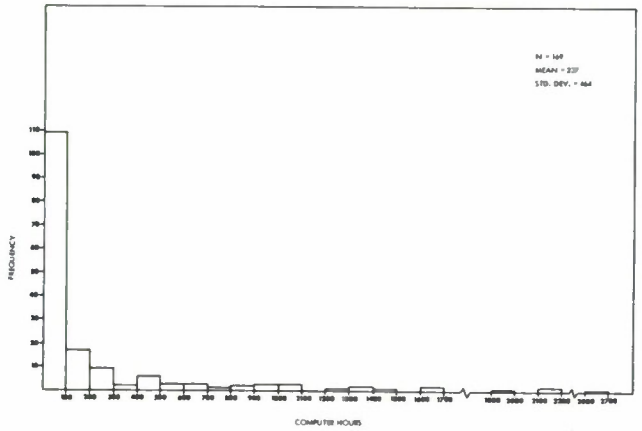


ESTIMATED VERSUS ACTUAL

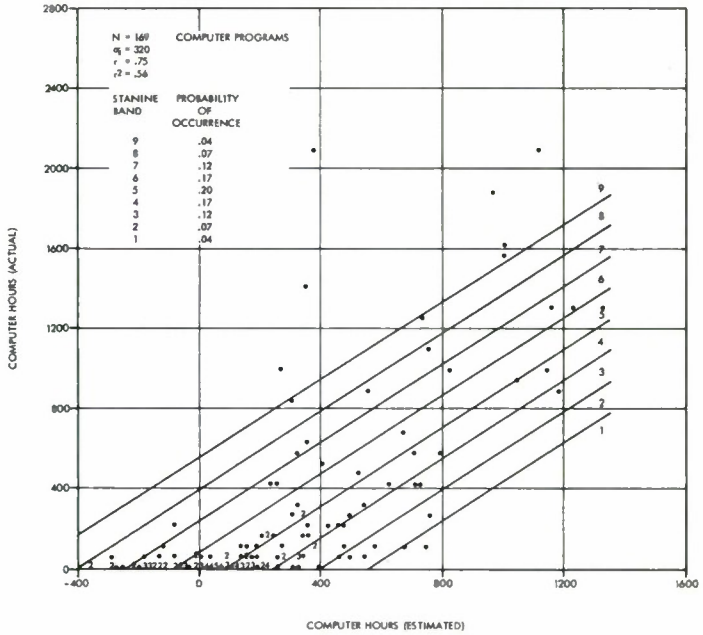
TABLE XX :	DATA BASE: TOTAL SAMPLE N = 169	RESOURCE: COMPUTER HOURS

TOTAL COMPUTER HOURS = Y_2 , WHERE:

$$Y_2 = +80.0 + 105X_8 + 2.70X_{35} + 21.2X_{37} + .158X_{46} + 10.8X_{47.1} - 6.85X_{47.3} + 292X_{48.3} - 63.3X_{52} + 266X_{56} + 3.59X_{57} - 3.95X_{64} + 240X_{72} - 173X_{76}$$



HISTOGRAM OF COMPUTER HOURS

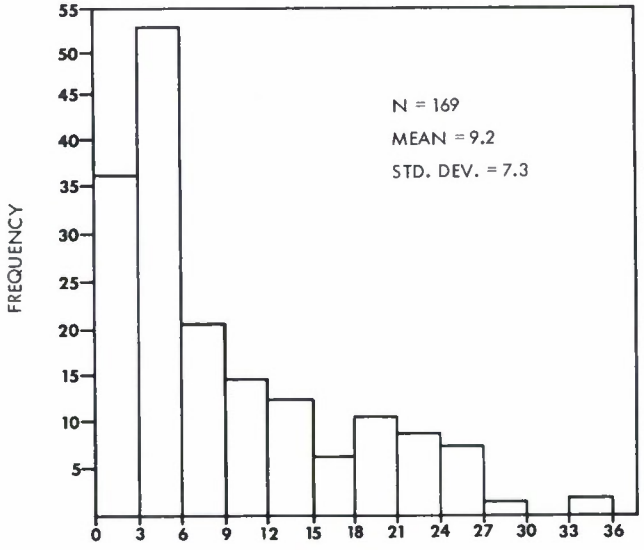


ESTIMATED VERSUS ACTUAL

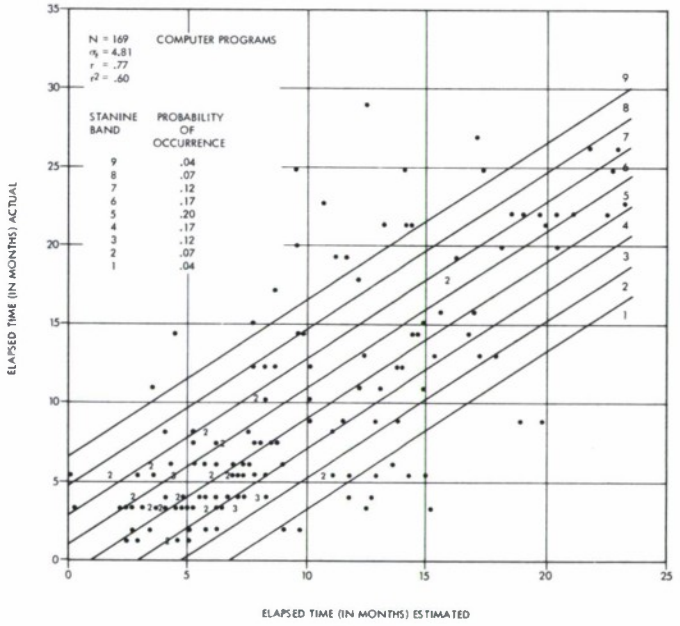
TABLE XXT :	DATA BASE: TOTAL SAMPLE N = 169	RESOURCE: MONTHS ELAPSED

TOTAL MONTHS ELAPSED = Y_3 , WHERE:

$$Y_3 = +1.31 + 1.31X_8 + .020X_{19} + .845X_{37} + 2.37X_{40} + .037X_{41} + .098X_{47.1} - .079X_{47.3} + .860X_{48.3} + 2.05X_{51} - 2.65X_{65} + 3.63X_{72} + .039X_{75} - 1.86X_{76}$$



HISTOGRAM OF MONTHS ELAPSED

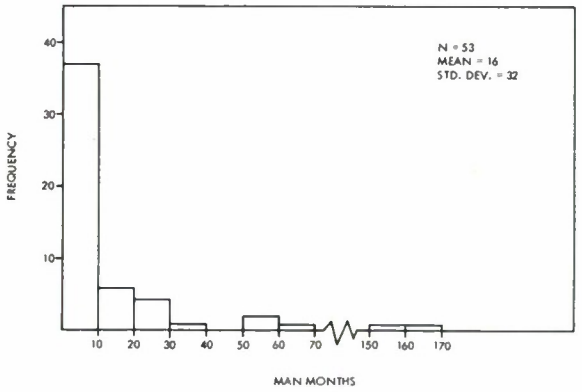


ESTIMATED VERSUS ACTUAL

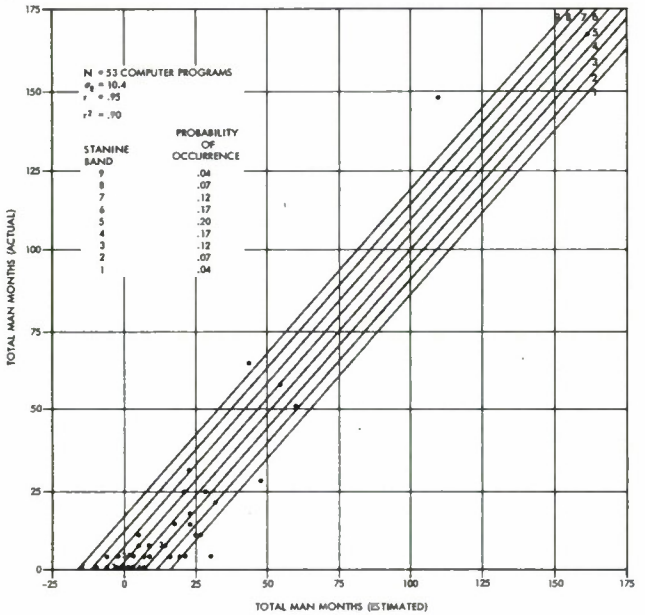
TABLE XXII :	DATA BASE: MEDIUM COMPUTER SUBSAMPLE N = 53	RESOURCE: MAN MONTHS

TOTAL MAN MONTHS = Y_1 , WHERE:

$$Y_1 = -47.3 + 5.73X_2 + .256X_4 + 3.09X_6 + 6.90X_8 + 4.59\log_{10}X_{18} + .159X_{41} + 20.0X_{48.3} - 5.46X_{48.5} + 17.2X_{53} + 14.7X_{54} + 4.68X_{56} + .010X_{64} + 12.5X_{71} + 1.06X_{75}$$



HISTOGRAM OF MAN MONTHS



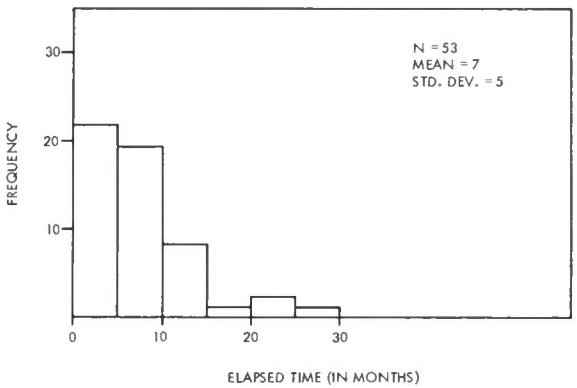
ESTIMATED VERSUS ACTUAL

Medium Computer Subsample consists of those machines whose monthly rental price, or equivalent purchase cost, is greater than $100,000 but less than $750,000.

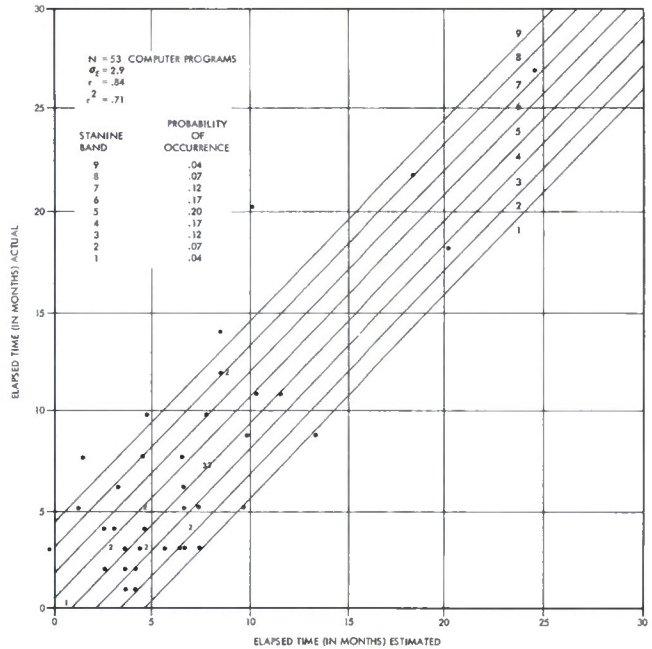
TABLE XXIII :	DATA BASE: MEDIUM COMPUTER SUBSAMPLE N = 53	RESOURCE: MONTHS ELAPSED

TOTAL MONTHS ELAPSED = Y_3 , WHERE:

$$Y_3 = -1.36 - 150X_4 + 1.54X_8 + .023X_{19} + .765X_{37} + .617X_{40} + .027X_{41} + .812X_{51} + 1.83X_{54} + 1.62X_{56} - 1.23X_{76}$$



HISTOGRAM OF MONTHS ELAPSED



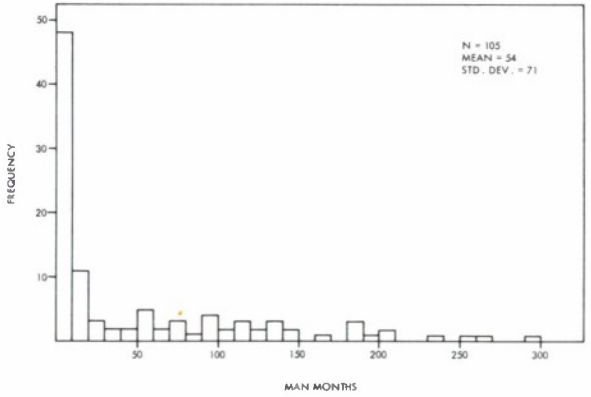
ESTIMATED VERSUS ACTUAL

The Medium Computer Subsample consists of those machines whose monthly rental price, or equivalent purchase cost, is greater than $100,000 but less than $750,000.

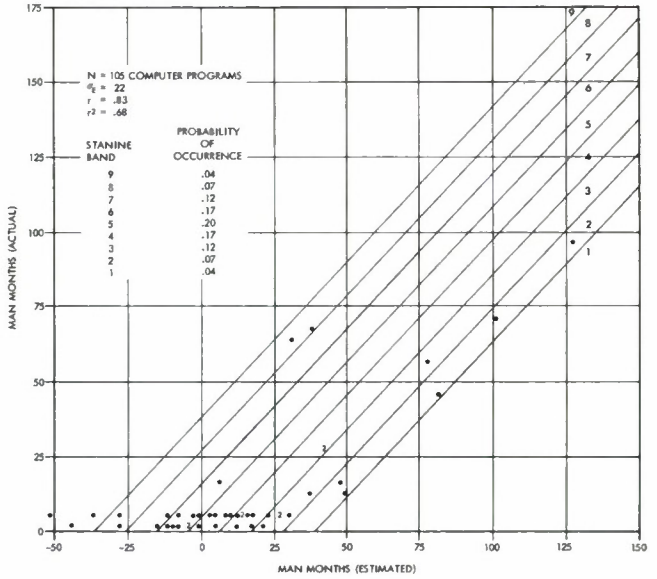
TABLE XXIV :	DATA BASE: LARGE COMPUTER SUBSAMPLE N= 105	RESOURCE: MAN MONTHS

TOTAL MAN MONTHS = Y_1 , WHERE:

$$Y_1 = +.049 + 15.2X_6 - .236X_{25} + .528X_{30} + 4.50X_{37} + .091X_{46} - 17.5X_{48.1} + 25.1X_{51} + 22.0X_{54} + 26.0X_{56} - .251X_{64} - 14.9X_{65} + 10.4X_{74}$$



HISTOGRAM OF MAN MONTHS

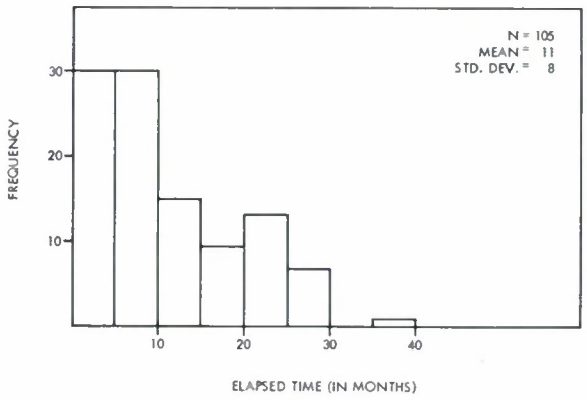


ESTIMATED VERSUS ACTUAL

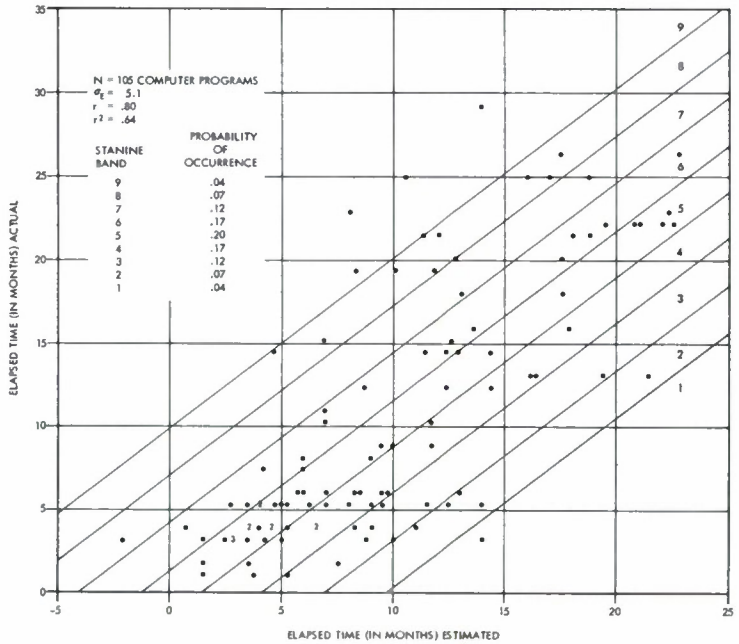
TABLE XXV:	DATA BASE: LARGE COMPUTER SUBSAMPLE N = 105	RESOURCE: MONTHS ELAPSED

MONTHS ELAPSED = Y_3 , WHERE:

$$Y_3 = +2.85 + 1.64X_2 + .862X_8 + .021X_{19} + 1.04X_{37} + 2.68X_{40} - 2.03X_{48.1} + 1.95X_{48.3} + 2.44X_{51} - 3.63X_{65} + .076X_{75} - 2.46X_{76}$$



HISTOGRAM OF MONTHS ELAPSED



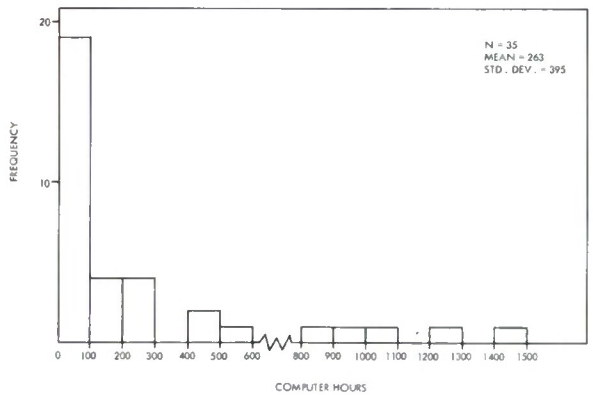
ESTIMATED VERSUS ACTUAL

The Large Computer Subsample consists of those machines whose monthly rental price, or equivalent purchase cost, is $750,000 or greater.

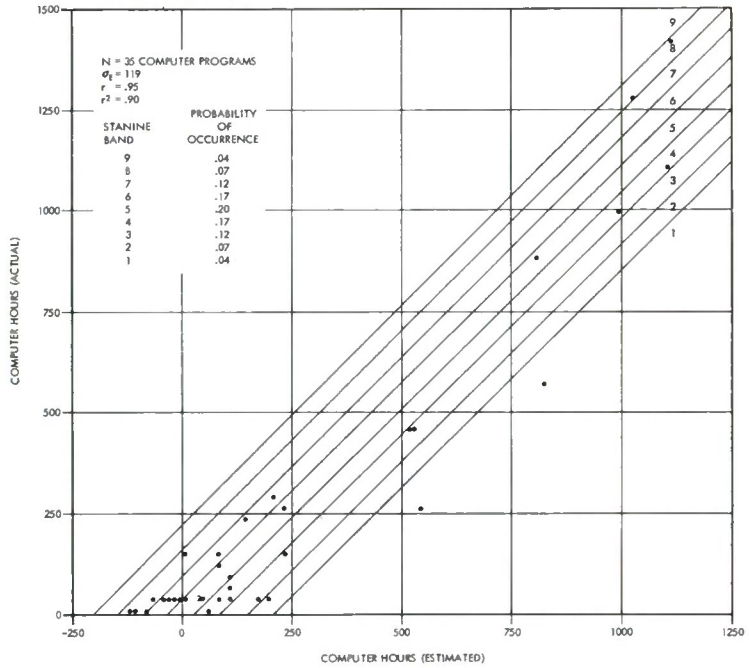
TABLE XXVI :	DATA BASE: TYPE OF PROGRAM (OTHER) SUBSAMPLE N= 35	RESOURCE: COMPUTER HOURS

TOTAL COMPUTER HOURS = Y_2 , WHERE:

$$Y_2 = -986 + 205X_{40} + 10.3X_{41} + 441X_{42} + .191X_{46} + 8.04X_{47.1} + 4.23X_{55} + 10.0X_{57} + 4.10X_{63} - 1.85X_{64} + 140X_{72}$$



HISTOGRAM OF COMPUTER HOURS

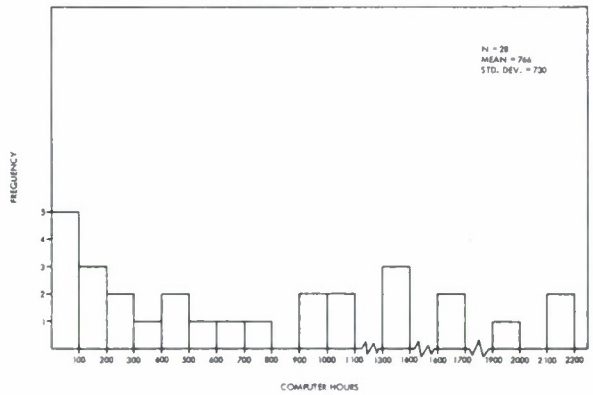


ESTIMATED VERSUS ACTUAL

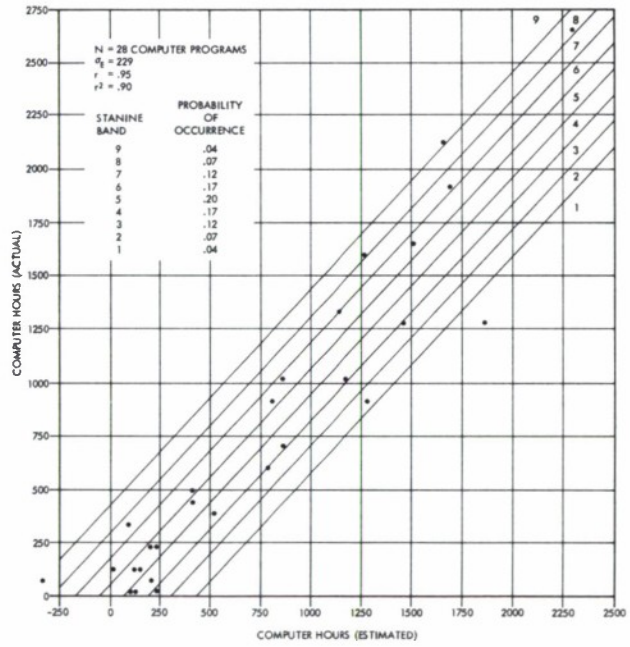
TABLE XXVII :	DATA BASE: TYPE OF PROGRAM (UTILITY) SUBSAMPLE N = 28	RESOURCE: COMPUTER HOURS

TOTAL COMPUTER HOURS = Y_2 , WHERE:

$$Y_2 = -756 + 212X_3 + 128X_{17} + 7.34X_{35} + 608X_{42} - 333X_{52} + 538X_{54} + 258X_{56} + 10.5X_{57} + 12.6X_{62} - 243X_{65}$$



HISTOGRAM OF COMPUTER HOURS



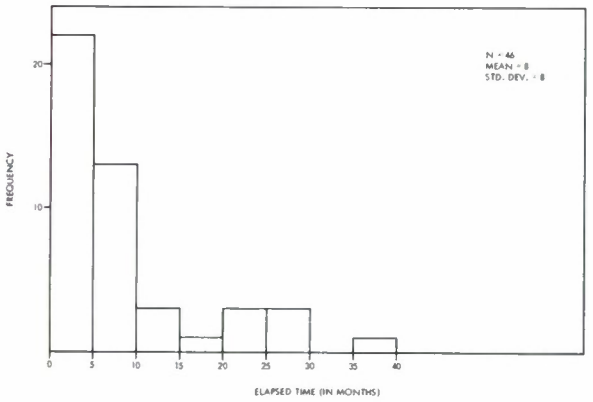
ESTIMATED VERSUS ACTUAL

The Utility Subsample consists of those programs that were written to support other programming activities. This subsample includes compilers, debugging aids and FIX routines.

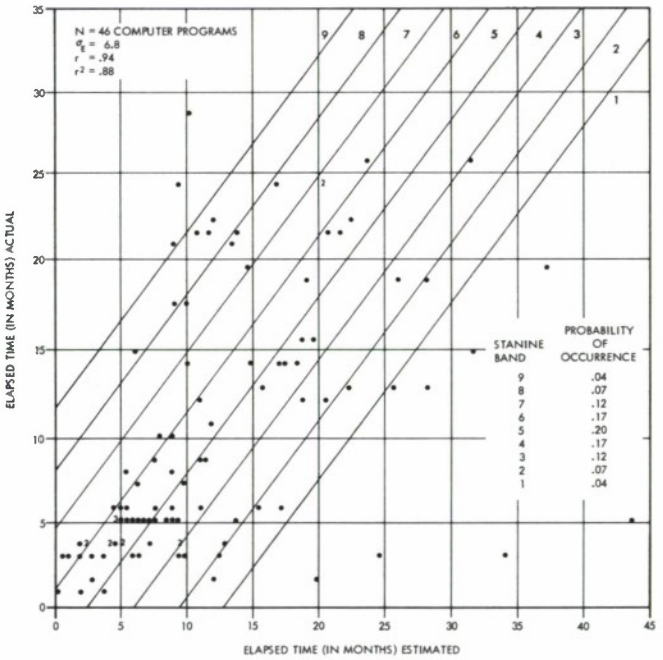
TABLE XXVIII :	DATA BASE: POL SUBSAMPLE N = 46	RESOURCE: MONTHS ELAPSED

TOTAL MONTHS ELAPSED = Y_3 , WHERE:

$$Y_3 = -2.88 + 1.83X_1 + 1.85X_2 + .297X_8 + .043X_{19} + .043X_{30} - 1.06X_{48.1} - .047X_{61} - 4.53X_{65} + .073X_{75} + 8.79X_{74}$$



HISTOGRAM OF MONTHS ELAPSED



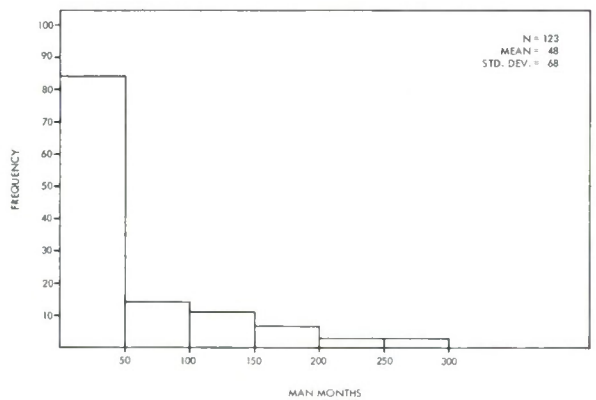
ESTIMATED VERSUS ACTUAL

The POL Subsample consists of those programs that were written in a procedure-oreiented language such as FORTRAN, COBOL, JOVIAL, etc.

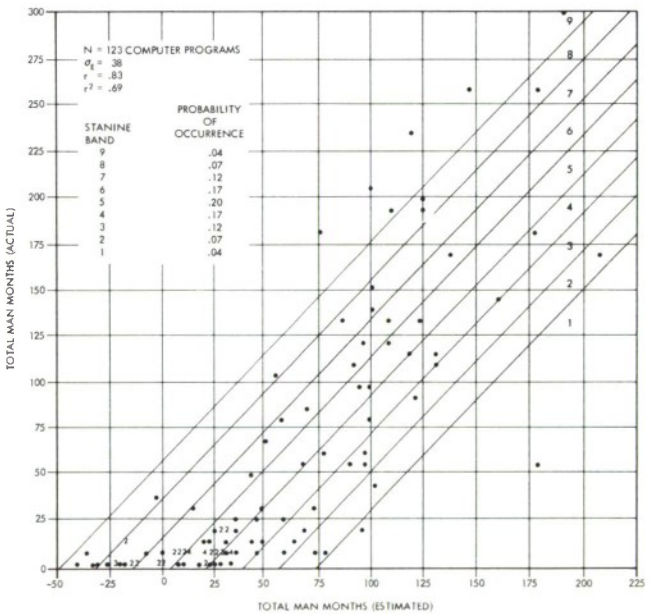
TABLE XXIX :	DATA BASE: MOL SUBSAMPLE N= 123	RESOURCE: MAN MONTHS

TOTAL MAN MONTHS = Y_1 , WHERE:

$$Y_1 = -32.9 + 1.28X_5 + 7.47X_8 + .466X_{35} + 10.1X_{40} + .329X_{41} + .058X_{46} - 5.94X_{48.1} + 8.93X_{51} + 41.4X_{53} + 14.9X_{54} + 20.7X_{56} + .699X_{57} - .104X_{64} - 8.00X_{65} + 1.04X_{75}$$



HISTOGRAM OF MAN MONTHS



ESTIMATED VERSUS ACTUAL

The MOL Subsample consists of those programs that were written in a machine-oriented language such as FAP, GAP, etc.

X_1 - Vagueness of Design Requirements Definition. Coded: direct translation of (simple) manual tasks to automatic functions = 0; only a few new functions to be automated, with requirements relatively clear and precise = 1; many new functions to be automated, requirements relatively precise = 2; many ill-defined and unstructured functions to be automated = 3.

X_2 - Innovation Required. Coded: yes = 1; no = 0.

X_3 - Lack of Knowledge of Operational Requirements. Coded: in great detail = 0; in broad outline = 1; only vaguely = 2.

X_4 - Number of Organizational Users. Coded: minimum value of 1.

X_5 - Number of ADP Centers. Coded: minimum value of 1.

X_6 - Complexity of Program System Interface. Coded: more than 50% of design effort devoted to data transfer problems to or from the program data point = 2; between 10% and 50% effort to data transfer problems = 1; less than 10% = 0.

X_8 - Stability of Design. Coded: initial design carried through without change = 0; few changes to initial program design = 1; frequent changes to program design = 2; initial program design almost completely revised = 3.

X_{17} - Number of Conditional Branches. Coded in thousands.

X_{18} - Number of Words in the Data Base. Coded in thousands.

X_{19} - Number of Classes of Items in the Data Base. Coded in raw score.

X_{25} - Percent Clerical Instructions. Coded in percent.

X_{26} - Percent Mathematical Instructions. Coded in percent.

X_{30} - Percent Information Storage and Retrieval Functions. Coded in percent.

X_{35} - Percent Generation Functions. Coded in percent.

X_{37} - Frequency of Operation. Coded: not applicable = 0; less than 1/month, more than 1/month and less than 1/week = 2; more than 1/week and less than 1/day = 3; daily = 4; utility or on-line (including compilers) = 5.

X_{40} - Stringent Timing Requirements. Coded: schedule restrictions imposed = 1; none imposed = 0.

X_{41} - Number of Subprograms. Coded in raw score.

X_{42} - Programming Language. Coded: MOL = 1; POL = 0.

X_{46} - External Documentation. Coded: number of pages written for, or distributed to, customers.

$X_{47.1}$ - Total Number of External Document Types Written.

$X_{47.3}$ - Total Number of Internal Document Types Written.

$X_{48.1}$ - Business, $X_{48.3}$ - Utility } Coded: as mutually exclusive binary variables; i.e., programs classified as business application = 1; remaining applications = 0.

$X_{48.5}$ - Stand Alone. Coded: yes = 1; no = 0.

X_{51} - First Program on Computer: Coded: yes = 1; no = 0.

X_{52} - Average Turnaround Time. Coded: less than 2 hours = 0; 2 to 11 hours = 1; 12 to 24 hours = 2; greater than 24 hours = 3.

X_{53} - ADP Components Developed Concurrently. Coded: components developed concurrently = 1; components not developed concurrently = 0.

X_{54} - Special Display Equipment: Coded: special display equipment used = 1; not used = 0.

X_{55} - Core Capacity.

X_{56} - Random Access Device Used. Coded: use of such storage = 1; such storage not used = 0.

X_{57} - Number of Bits per Word.

X_{61} - Percent Senior Programmers. Coded: the ratio of the total number of senior or systems programmers to the total number of programmers assigned to the project.

X_{62} - Average Programmer Experience with Language. Coded in months.

X_{63} - Average Programmer Experience with Application. Coded in months.

X_{64} - Percent Programmers Participating in Program Design.

X_{65} - Personnel Continuity. Coded: number of personnel working for the duration of the project, divided by the maximum number assigned at any one time.

X_{71} - Program Developed at Site Other Than the Operational Installation. Coded: yes = 1; no = 0.

X_{72} - Different Computers for Programming and Operation. Coded: yes = 1; no = 0.

X_{74} - Number of Locations for Program Data Point Development.

X_{75} - Number of Man Trips.

X_{76} - Program Data Point Developed by Military Organization. Coded: yes = 1; no = 0.

(page 90 blank)

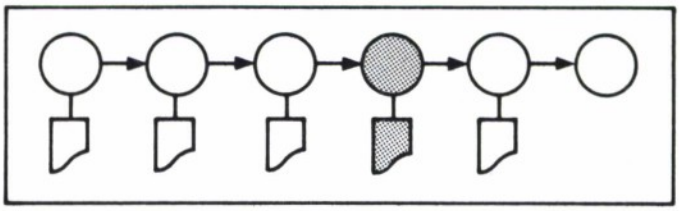


TABLE XXX: ACTIVITY DEFINITION

ACTIVITY: INFORMATION PROCESSING SYSTEM INTEGRATION TEST

DESCRIPTION: This activity covers all work necessary to test the performance of the computer program within the total system at the operational facility under realistic ("live") operating conditions.

(AFSCM 375 context: This activity occurs in the Acquisition Phase and is equivalent to Category II testing.)

TASKS: Conduct Test. Within the requirements of the plans for program testing, conduct a sequence of tests of the total operational system, receiving actual data from and transmitting actual data to other subsystems and components that constitute the system.

Analysis of Test Results. Determine if the system meets specifications, and study operations for any evidence of potential difficulties. Coordinate with other subsystem and component developers to isolate sources of poor system performance.

Initiate Modifications to Computer Programs. Correct errors in programs and associated documentation. Design and implement feasible changes to meet specified performance requirements. This involves work in the earlier activities.

Documentation of Test Results. Prepare appropriate compliance documentation to certify successful tests. If problem areas arise, document the evidence for use in the modification process. Identify potential improvements that could be made to the total system.

TABLE XXX : ACTIVITY DEFINITION (CONT'D.)

PRINCIPAL INPUTS:	Information system test plans. Information system specifications. Program system design specifications. Program system symbolic deck, binary deck, listing. Operating system data. Descriptions of the design and operation of other subsystems and components.
PRINCIPAL OUTPUTS:	Program system test compliance documentation. Modifications including error corrections and changes for the computer programs and their documentation. Recommendations for future modifications.

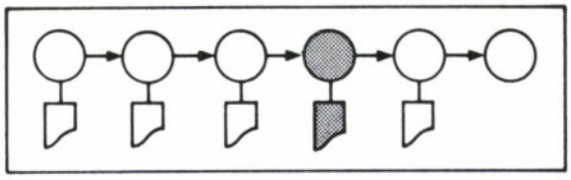


TABLE XXXI: COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_4 - Number of organizational users	+	+	+			
X_5 - Number of ADP centers	+	+	+			
X_8 - Stability of design	+	+	+			
X_{84} - Number of functions in the system	+		+			
X_{85} - Number of system components	+		+			
X_{86} - Number of system components not off-the-shelf	+		+			
Program Design & Production						
X_{10} - Total object instructions delivered		+				23
X_{17} - Number of conditional branches		+				
X_{18} - Number of words in the data base		+				23
X_{19} - Number of classes of items in the data base		+				23
X_{20} - Number of input message types		+				23
X_{21} - Number of output message types		+				23
X_{22} - Number of input variables		+				23

*NOTE: IMPACT IS INDICATED BY:

+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SIGN = NO PRESUMED DIRECTION

(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE XXXI : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
					OTHER	
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	DESCRIPTIVE	ANALYTICAL
X_{23} - Number of output variables		+				
X_{36} - Average operate time		+				
X_{42} - Programming language						
$X_{45.1}$ - Internal documentation written	+	+	+			
$X_{45.2}$ - Internal documentation available	-	-	-			
X_{46} - External documentation	+	+	+			
$X_{47.1}$ - Total number of external document types written	+		+			
$X_{47.2}$ - Total number of internal document types available	-	-	-			
$X_{47.3}$ - Total number of internal document types written	+		+			
X_{48} - Type of program						
Data Processing Equipment						
X_{51} - First program on computer	+	+	+			
X_{52} - Average turnaround time			+			
X_{53} - ADP components developed concurrently	+	+	+			
X_{54} - Special display equipment	+	+	+			
X_{55} - Core capacity		-				23
X_{56} - Random access device used		+				
X_{57} - Number of bits per word		-				
X_{58} - Memory access time		+				
X_{59} - Machine add time		+				
X_{60} - Computer cost		-				23

TABLE XXXI : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Personnel						
X_{61} - Percent senior programmers	-	-	-			
X_{62} - Average programmer experience with language	-	-	-			
X_{63} - Average programmer experience with application	-	-	-			23
X_{65} - Personnel continuity	-	-	-			
X_{87} - Percent senior analysts	-	-	-			
X_{92} - Personnel turnover	+	+	+			
Development Environment						
X_{67} - Lack of management procedures	+	+	+			
X_{68} - Number of agencies concurring in design	+	+	+			
X_{70} - Computer operated by agency other than program developer	+	+	+			
X_{71} - Program developed at site other than the operational installation	+	+	+			
X_{72} - Different computers for programming and operation	+	+	+			
X_{79} - Security classification level	+		+			
X_{88} - Quality of resource documents	-	-	-			
X_{89} - Availability of special tools	-	-	-			
X_{90} - Degree of standardization in policy and procedures	-	-	-			
X_{91} - Number of official reviews of documents	+		+			

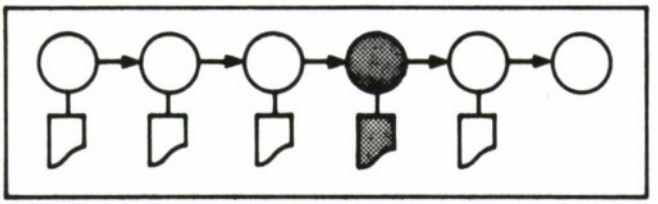


TABLE XXXII : PLANNING FACTORS

TYPE: PERCENT OF OTHER ITEM

SUBJECT: COSTING SYSTEM INTEGRATION TEST

Page 57 of Reference (18) shows the following average percentages for the relative division of costs:

Activity	% of Total Project Costs
Analysis and Design	34.5
Program Coding and Checking	18
Checkout and Test	47.5

The activity labeled Checkout and Test would include the Information System Integration Test activity defined in this Handbook. It is estimated that integration test would range from 0% to 30% of the total cost for computer programming depending upon (1) the number of components and subsystems in the total information system (2) how new these are and (3) how thoroughly any new component or subsystem had been tested prior to the integration test.

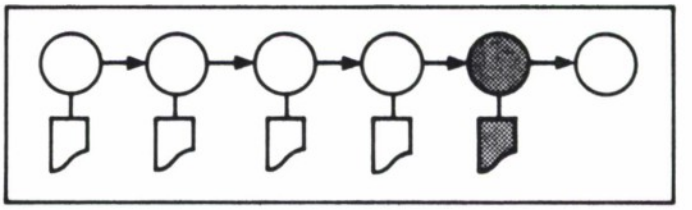


TABLE XXXIII:	ACTIVITY DEFINITION
ACTIVITY:	INFORMATION PROCESSING SYSTEM INSTALLATION AND TURNOVER
DESCRIPTION:	The purpose of the turnover step is to help the user demonstrate, at his own operational site, that the computer program system will operate as specified, and to support the user with documentation, advice, and guidance, and troubleshooting during the initial period of system operation. (AFSCM 375 context: Information System Installation and Turnover occurs in the Acquisition-Operational Overlap Phase, beginning with the installation of the information processing contract end-item at an operational site other than the Category II site, and ending with turnover of the information system to the user at the last operational site.)
TASKS:	Verify Program System Specifications. Verify the accuracy and completeness of program system specifications and other documents. Produce any modifications needed. Prepare User Documentation. Determine user documentation requirements and, if necessary, issue specifications for this documentation. Prepare documentation production plan. Perform the technical writing and editing necessary to produce user documentation, coordinate the production of material by all contributors, and verify the adequacy and accuracy of the results. Obtain user approval and concurrence on documentation. Advise User on Data Conversion. Assist user in preparing data collection and conversion plans; advise on the technical aspects of the task and the adequacy of results.

TABLE XXXIII: ACTIVITY DEFINITION (CONT'D.)

Develop User Training Plan. Determine training requirements including: characteristics of those to be trained; present and future duties; extent and content of training required. Assist user in developing a curriculum and schedule. Prepare course materials and visual aids required.

Conduct Training Program. Conduct classes, briefings, or demonstrations to train user personnel to interpret and prepare inputs and outputs, and to control and maintain the computer and the program system.

Conduct Demonstration Test. Produce or assist in the production of system test material required for the demonstration test. Brief participating personnel on the objectives and procedures of the demonstration. Dry-run the test. Conduct the test jointly with user personnel, and if necessary document the results. Conduct debriefing sessions after demonstration tests.

Assist in Operational Shakedown. Monitor initial period of operation of the system to detect and correct malfunctions and inefficiencies, and to evaluate performance. Establish procedures for the user to report suspected program malfunctions. Identify reported malfunctions as either:

(1) error in programming (requirements understood but incorrectly implemented)

(2) error in logic (requirements not understood)

(3) operator error

(4) equipment failure

and take corrective action.

PRINCIPAL INPUTS:

System functional description and specifications

Program system design documentation

Design change documents

Program flow charts

TABLE XXXIII: ACTIVITY DEFINITION (CONT'D.)

	Index to and descriptions of computer program documents
	Source program listings
	Object programs
	Prior system documentation
	Data base design and data definition documents
PRINCIPAL OUTPUTS:	User documentation
	User training plan and material
	System test material

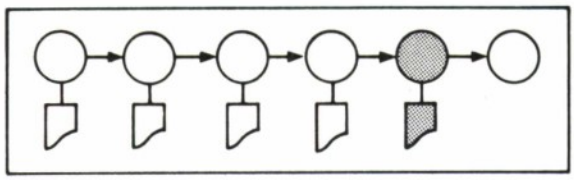


TABLE XXXIV: COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_1 - Vagueness of design requirements definition	+	+	+			
X_2 - Innovation required	+	+	+			
X_3 - Lack of knowledge of operational requirements	+	+	+			
X_4 - Number of organizational users	+	+	+			
X_5 - Number of ADP centers	+	+	+			
X_6 - Complexity of program system interface	+	+	+			
X_9 - On-line requirements						
X_{78} - Complexity of system interface with other systems	+	+	+			
X_{83} - Degree of system change expected during operations	+	+	+			
X_{84} - Number of functions in the system	+	+	+			
X_{85} - Number of system components	+	+	+			
X_{86} - Number of system components not off-the-shelf	+	+	+			

*NOTE: IMPACT IS INDICATED BY:

+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SIGN = NO PRESUMED DIRECTION

(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE XXXIV : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

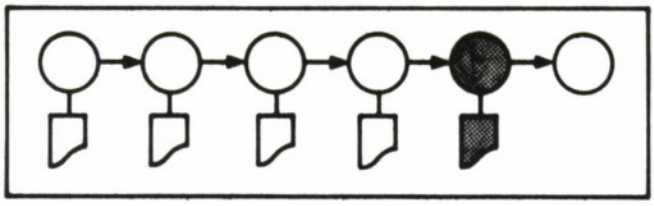
FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Program Design & Production						
X_{10} - Total object instructions delivered	+	+	+			23
X_{13} - Total source instructions written	+	+	+			23
X_{14} - Percent source instructions written in POL	-	-	-			
X_{17} - Number of conditional branches	+	+	+			
X_{18} - Number of words in the data base	+	+	+			23
X_{19} - Number of classes of items in data base	+	+	+			23
X_{20} - Number of input message types	+	+	+			23
X_{21} - Number of output message types	+	+	+			23
X_{22} - Number of input variables	+	+	+			23
X_{23} - Number of output variables	+	+	+			
X_{24} - Number of words in tables, and constants not in data base	+	+	+			
X_{36} - Average operate time		+				
X_{41} - Number of Subprograms	+	+	+			
X_{42} - Programming language						
$X_{45.1}$ - Internal documentation written	+	+	+			
$X_{45.2}$ - Internal documentation available	-	-	-			
X_{46} - External documentation	+	+	+			
$X_{47.1}$ - Total number of external document types written	+	+	+			
$X_{47.2}$ - Total number of internal document types available	-	-	-			

TABLE XXXIV : COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
$X_{47.3}$ - Total number of internal document types written	+	+	+			
X_{48} - Type of program						
X_{93} - Output volume	+	+	+			23
X_{94} - Input volume	+	+	+			23
Data Processing Equipment						
X_{50} - Developmental computer used						
X_{53} - ADP components developed concurrently	+	+	+			
X_{54} - Special display equipment	+	+	+			
X_{56} - Random access device used		+				
X_{57} - Number of bits per word		-				
X_{58} - Memory access time		+				
X_{59} - Machine add time		+				
X_{60} - Computer cost	+	+	+			23
Personnel						
X_{61} - Percent senior programmers	-	-	-			
X_{62} - Average programmer experience with language	-	-	-			
X_{63} - Average programmer experience with application	-	-	-			23
X_{65} - Personnel continuity	-	-	-			
X_{87} - Percent senior analysts	-	-	-			
X_{92} - Personnel turnover	+	+	+			

TABLE XXXIV: COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Development Environment						
X_{67} - Lack of management procedures	+	+	+			
X_{68} - Number of agencies concurring in design	+	+	+			
X_{69} - Customer inexperience	+	+	+			
X_{70} - Computer operated by agency other than program developer	+	+	+			
X_{71} - Program developed at site other than the operational installation	+	+	+			
X_{72} - Different computers for programming and operation	+	+	+			
X_{79} - Security classification level	+		+			
X_{88} - Quality of resource documents	-	-	-			
X_{89} - Availability of special tools	-	-	-	104		
X_{90} - Degree of standardization in policy and procedures	-	-	-			
X_{91} - Number of official reviews of documents	-	-	-			



ADDITIONAL COMMENTS ON SELECTED COST FACTORS

X_{89} - The Availability of Special Tools. In the information system turnover activity, particularly in the data conversion task, the availability of special tools could be considered to include special input devices such as magnetic ink character recognition, or optical character recognition (51). The economics of such devices are obviously a function of the volume of data to be processed (note that for this activity the job is to build the working data base, not process incoming data on an ongoing basis). For optical character recognition, one study estimated the breakeven point for the Recognition Equipment, Inc., Electronic Retina Computing Reader to be a monthly data volume of about a half million card equivalents, where the alternative is keypunching with 100 percent verification (47).

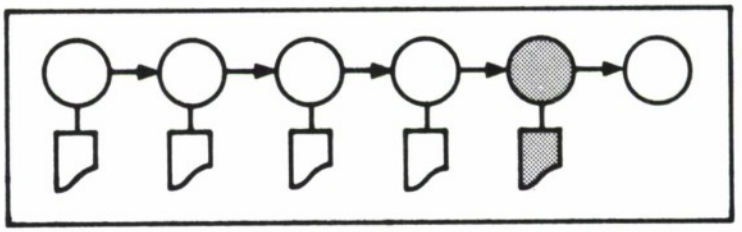


TABLE XXXV: PLANNING FACTORS

TYPE: VARIOUS SUBJECT: COSTING DEMONSTRATION TESTS

TASK	COSTING FORMULA
Conduct Demonstration Test	Final preparation and actual conduct of the test for a program system of moderate size should take an elapsed time of about one week. One or more dry runs, especially if associated with operator training, may add one or more weeks to the schedule. Estimate about two man weeks for a program system of about 10,000 to 20,000 instructions.
Analysis of Test Results	Allow about one man week for system of about 10,000 to 20,000 instructions.
Documentation of Test Results	Allow two man weeks for drafting test report.

NOTE: When a formal program system demonstration test is not required, as is sometimes the case when programs are operated and developed by the same organization, this activity is covered by the "Functional Test of Complete Program System" task in the previous activity, computer program design, code, and test.

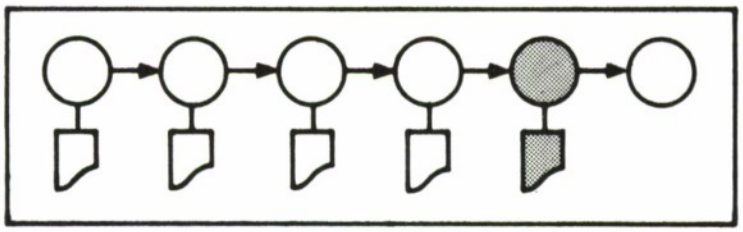


TABLE XXXVI: PLANNING FACTORS

TYPE: UNIT COST SUBJECT: ESTIMATION OF TASKS WITHIN ACTIVITY

TASK	COSTING FORMULA
Verify Program System Specifications	Technical review: 20 pages per man day Revise: 10 pages per man day Collect information: 2 days per document plus 2 hours per interview Type: 20 pages per man day Expect at least two drafts of revised specifications (18)
Prepare User Documentation	Outline: 2 man weeks per user's document Drafting rate: 3-5 pages per man day Technical review: 20 pages per man day Edit: 50 pages per man day Revise: 10 pages per man day Type: 20 pages per man day Illustrate: 2 illustrated pages per man day (18)
Data Conversion	Keypunching: Key Strokes Per Hour Including Set-Up (8) All alphabetic punching 5,000 Mixed alphabetic-numeric 4,000 Mostly numeric, little alphabetic 8,000 All numeric punching 10,000 Keypunching or key verification generally runs $1.00 per 1000 card per card column (56).

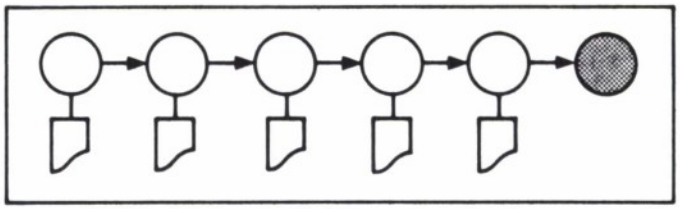


TABLE XXXVII: ACTIVITY DEFINITION

ACTIVITY: COMPUTER PROGRAM MAINTENANCE

DESCRIPTION: Computer program maintenance is the process of improving, changing, and correcting computer programs in an information system that is currently operational.

Program maintenance, including both revisions and error corrections, is needed throughout the life of the information system. Revisions are needed because operational requirements are continually changing during both the development and operation of the system. Although operational needs are projected during requirements analysis, in most cases they can be neither totally defined nor totally implemented in the imposed time schedules. Also, corrections must usually be made to the computer programs because errors and operational deficiencies not detected in the routine testing of the programs are usually discovered when the system becomes operational.

Since the need for improvement and support activities for the information system tends to be amorphous, system maintenance is often funded at a level the user can afford or is willing to spend rather than the level precisely required. Much of the work of program maintenance personnel must be devoted to the resolution of emergencies; a good share of the remainder, to modifications required by hard-to-predict environmental changes.

TABLE XXXVII: ACTIVITY DEFINITION (CONT'D.)

TASKS:	Develop a Maintenance Plan and Organization. Estimate level of maintenance activity required for the computer program system. Establish the amount of surplus resources desired to insure the proper handling of emergency situations. Determine responsibility and authority for the maintenance function. Procure and train necessary personnel. Provide Communications Between User and Computer Program Developer. Establish and maintain formal communications between the information system user, the computer program developer, and other interacting agencies. Establish Internal Communication Channels. Provide internal communication channels for processing information flow between various operational sites. Provide for periodic progress and activity reports. Establish Change Procedures. Establish procedures for installing error corrections and system retrofits into the operating computer program system. Provide procedures for updating computer program and information system documentation. Process System Design Changes. Whether design changes emanate from requirements changes, or the detection of program error, generally this task involves all of the preceding steps in the programming process. Provide User Support. Provide consulting support on management problems, information system hardware, data collection and conversion, and system operation, as required by the user.
PRINCIPAL INPUTS:	Computer program documentation, including design specifications and descriptions, source program listings, and user manuals. Design change requirements. Information system specifications. System malfunction reports.

TABLE XXXVII: ACTIVITY DEFINITION (CONT'D.)

PRINCIPAL OUTPUTS:	Revised program code.
	Updated listings and master tapes.
	Updated documentation.
	User advice.

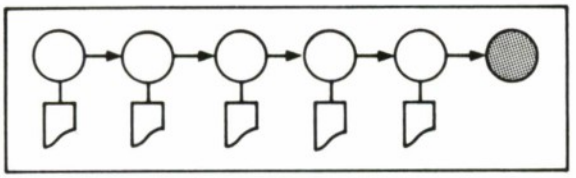


TABLE XXXVIII : COMPUTER PROGRAMMING COST FACTORS

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Requirements						
X_4 - Number of organizational users	+	+	+		63	
X_5 - Number of ADP centers	+	+	+			
X_6 - Complexity of program system interface	+	+	+		63	
X_9 - On-Line requirements	+				63	
X_{78} - Complexity of system interface with other systems	+	+	+		63	
X_{84} - Number of functions in the system	+	+	+			
X_{85} - Number of system components	+	+	+			
X_{86} - Number of system components not off-the-shelf	+	+	+			
X_{93} - Output volume	+	+	+			23
X_{94} - Input volume	+	+	+			23
Program Design & Production						
X_{10} - Total object instructions delivered	+	+	+		63	
X_{18} - Number of words in the data base	+	+	+			

*NOTE: IMPACT IS INDICATED BY:

+ = VARIES DIRECTLY
- = VARIES INVERSELY
NO SIGN = NO PRESUMED DIRECTION

(FOR DISCUSSION OF CODING, SEE GLOSSARY D)

TABLE XXXVIII: COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

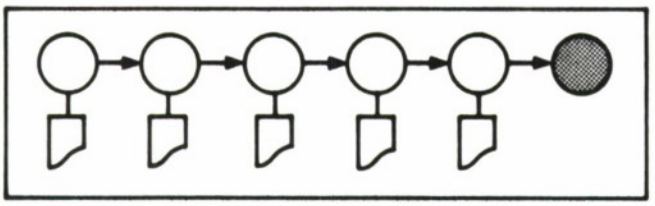
FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
					OTHER	
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	DESCRIPTIVE	ANALYTICAL
X_{19} - Number of classes of items in data base	+	+	+			
X_{20} - Number of input message types	+	+	+			23
X_{21} - Number of output message types	+	+	+			23
X_{22} - Number of input variables	+	+	+			23
X_{23} - Number of output variables	+	+	+			
X_{24} - Number of words in tables and constants not in data base	+	+	+			
X_{36} - Average operate time	+	+	+			
X_{37} - Frequency of operation	+	+	+	114		
X_{42} - Programming language				114	55	
$X_{45.1}$ - Internal documentation written	+	+	+			
$X_{45.2}$ - Internal documentation available	-	-	-			
X_{46} - External documentation required	+	+	+			
$X_{47.1}$ - Total number of external document types written	+	+	+			
$X_{47.2}$ - Total number of internal document types available	-	-	-			
$X_{47.3}$ - Total number of internal document types written	-	-	-		63	
X_{48} - Type of program						
Data Processing Equipment						
X_{51} - First program on computer	+	+	+			
X_{53} - ADP components developed concurrently	+	+	+			
X_{54} - Special display equipment	+	+	+			

TABLE XXXVIII: COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
Personnel						
X_{61} - Percent senior programmers	-	-	-			
X_{62} - Average programmer experience with language	-	-	-			
X_{63} - Average programmer experience with application	-	-	-			
X_{64} - Percent programmers participating in design	-	-	-			
X_{65} - Personnel continuity	+	+	+			
X_{66} - Maximum number of programmers assigned	+	+	-	114	63, 38	
X_{87} - Percent senior analysts	-	-	-			
X_{92} - Personnel turnover	+	+	+			
Development Environment						
X_{67} - Lack of management procedures	+	+	+		63	
X_{69} - Customer inexperience	+	+	+		63	
X_{70} - Computer operated by agency other than developer	+	+	+			
X_{71} - Program developed at site other than the operational installation	+	+	+			
X_{73} - Closed or open shop operation	+	+	-			
X_{79} - Security classification level	+	+	+			
X_{80} - Number of sources of system information	+	+	+			
X_{81} - Accessibility of system information	-	-	-			
X_{88} - Quality of resource documents	-	-	-		57	

TABLE XXXVIII: COMPUTER PROGRAMMING COST FACTORS (CONT'D.)

FACTOR (FOR DEFINITION & CODING, SEE GLOSSARY A)	*IMPACT ON INDICATED RESOURCE			REFERENCE		
	MAN MONTHS	COMP. HOURS	ELAPSED TIME	HANDBOOK PAGE	OTHER	
					DESCRIPTIVE	ANALYTICAL
X_{89} - Availability of special tools	-	-	-			
X_{90} - Degree of standardization in policy and procedures	-	-	-		63	
X_{91} - Number of official reviews of documents	-	-	-			



ADDITIONAL COMMENTS ON SELECTED COST FACTORS

X_{37} - Frequency of Operation. The primary rationale for believing that the frequency of operation affects maintenance programming resources is that (a) a high-frequency operation offers incentive to improve the efficiency of the program, and (b) there is a greater probability of discovering errors during production runs.

X_{42} - Programming Language. A major claim made for procedure-oriented languages, as compared with assembly languages, is that they reduce significantly the amount of effort needed for program production and maintenance. The basis for this claim is twofold: (a) a program written in procedure language is shorter (contains fewer steps) than an equivalent program written in an assembly language, and (b) a program written in a procedure language is easier to modify than one written in assembly language (55).

X_{66} - Maximum Number of Programmers Assigned. In the program maintenance step, a larger number of programmers may be assigned than actually required at any arbitrary point in time. The reason for this would be to insure an adequate response to emergency situations; that is, if the nature of a program was such that the cost of a delay in correcting an error may far exceed the cost of extra program maintenance personnel, added personnel may be used as a form of protection against this risk (63) wages are invested to buy short elapsed time during troubles.

On balance, larger programming organizations, if properly managed, need not be less efficient than their smaller counterparts (38).

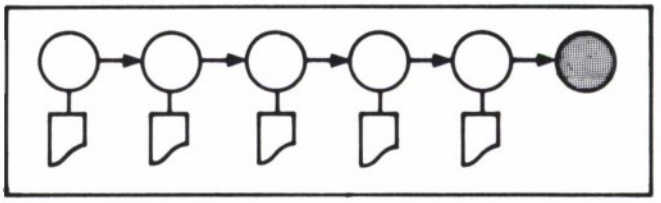
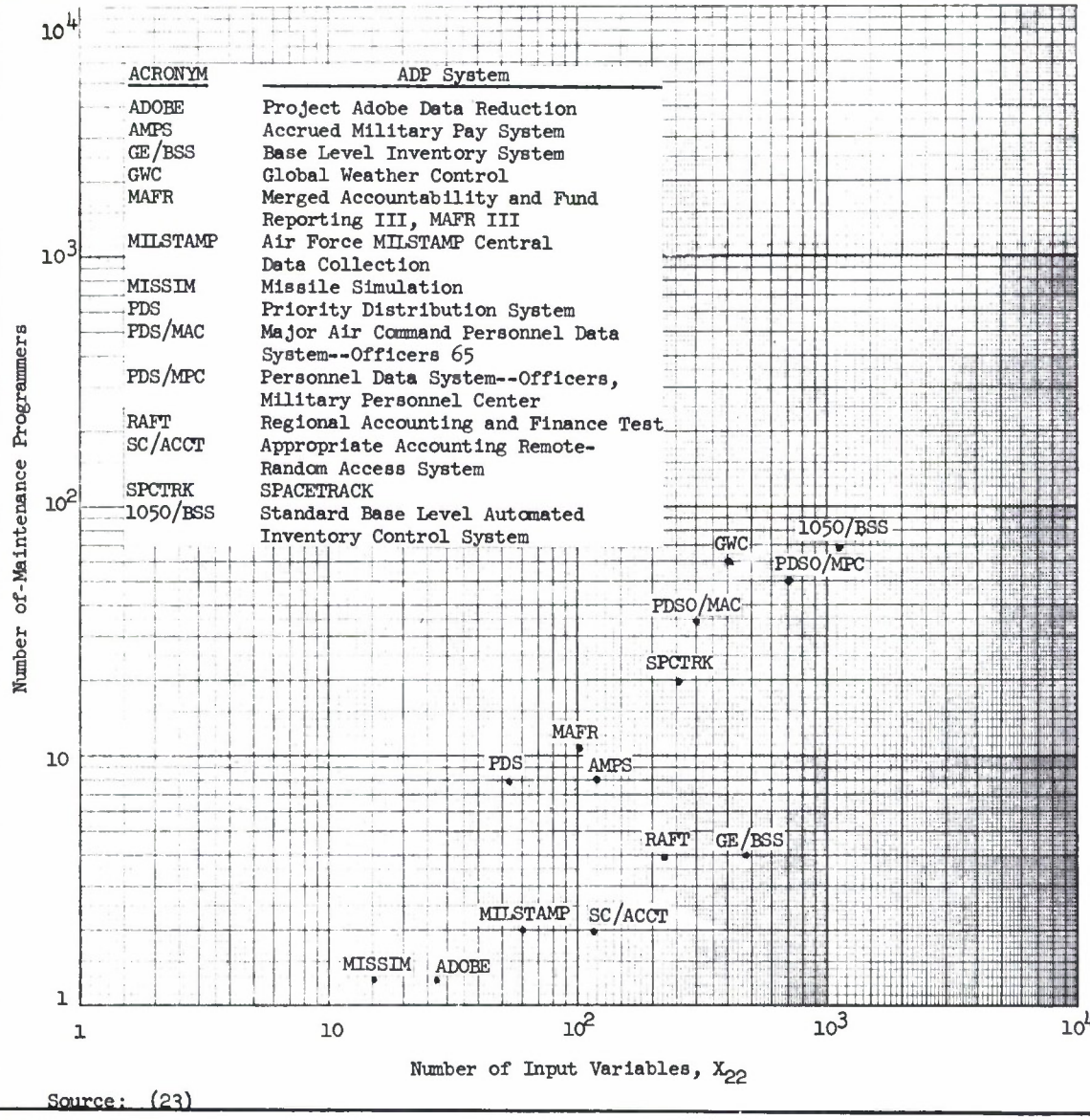


TABLE XXXIX: PLANNING FACTORS

TYPE: Unit Cost

SUBJECT: Estimating the Number of Maintenance Programmers

ACRONYM	ADP System
ADOBE	Project Adobe Data Reduction
AMPS	Accrued Military Pay System
GE/BSS	Base Level Inventory System
GWC	Global Weather Control
MAFR	Merged Accountability and Fund Reporting III, MAFR III
MILSTAMP	Air Force MILSTAMP Central Data Collection
MISSIM	Missile Simulation
PDS	Priority Distribution System
PDS/MAC	Major Air Command Personnel Data System--Officers 65
PDS/MPC	Personnel Data System--Officers, Military Personnel Center
RAFT	Regional Accounting and Finance Test
SC/ACCT	Appropriate Accounting Remote-Random Access System
SPCTRK	SPACETRACK
1050/BSS	Standard Base Level Automated Inventory Control System



Source: (23)

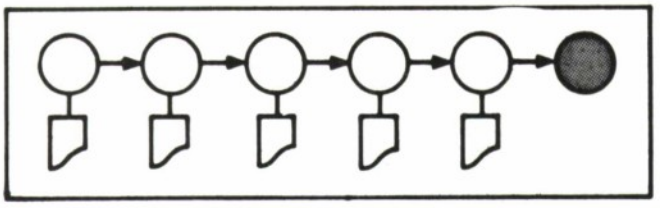
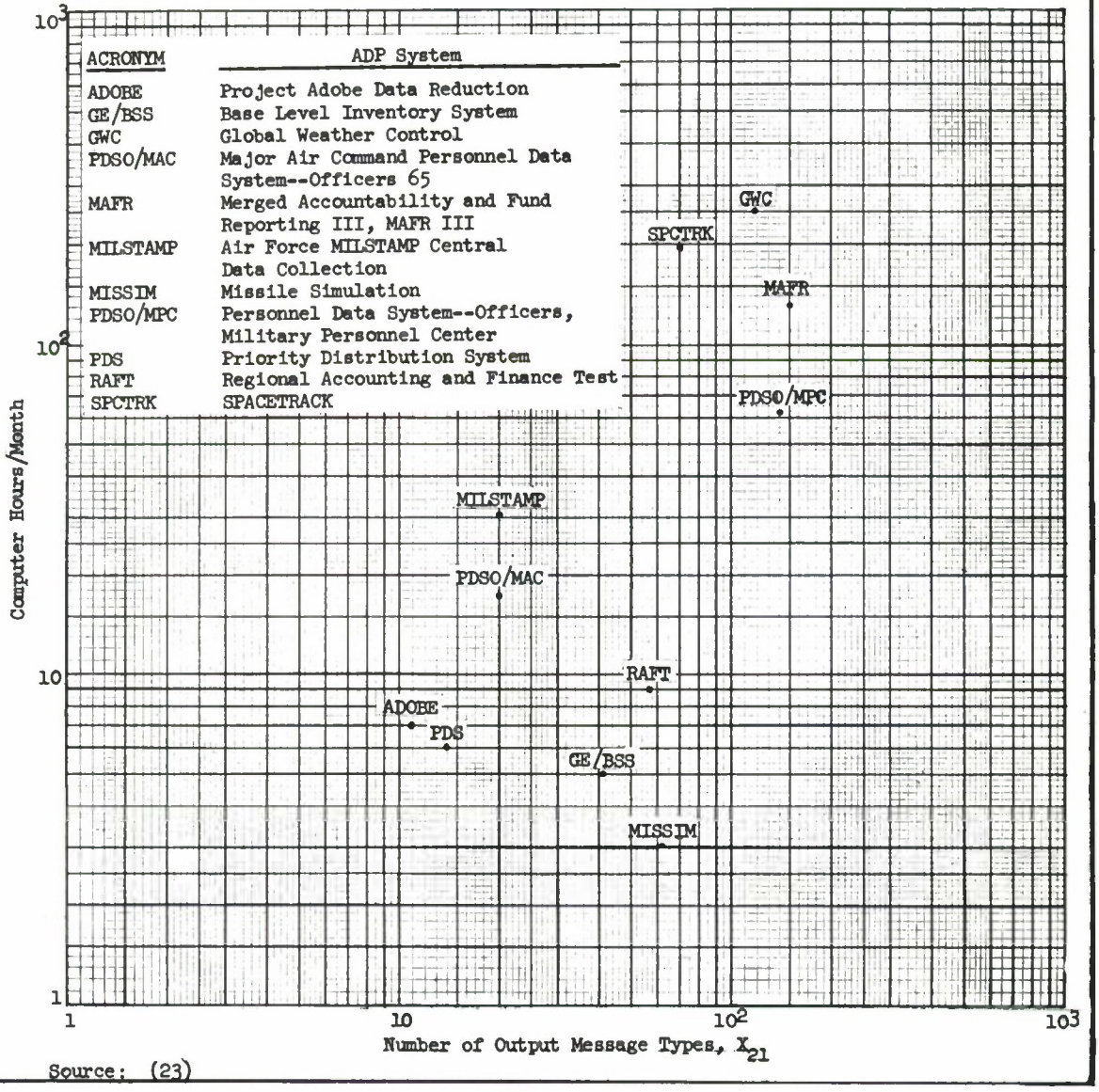


TABLE XL: PLANNING FACTORS

TYPE: Unit Cost

SUBJECT: Estimating Program Maintenance Computer Hours

ACRONYM	ADP System
ADOBE	Project Adobe Data Reduction
GE/BSS	Base Level Inventory System
GWC	Global Weather Control
PDSO/MAC	Major Air Command Personnel Data System--Officers 65
MAFR	Merged Accountability and Fund Reporting III, MAFR III
MILSTAMP	Air Force MILSTAMP Central Data Collection
MISSIM	Missile Simulation
PDSO/MPC	Personnel Data System--Officers, Military Personnel Center
PDS	Priority Distribution System
RAFT	Regional Accounting and Finance Test
SPCTRK	SPACETRACK



Source: (23)

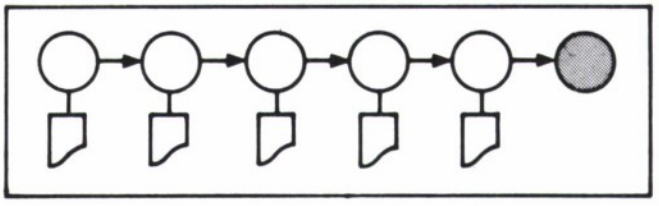
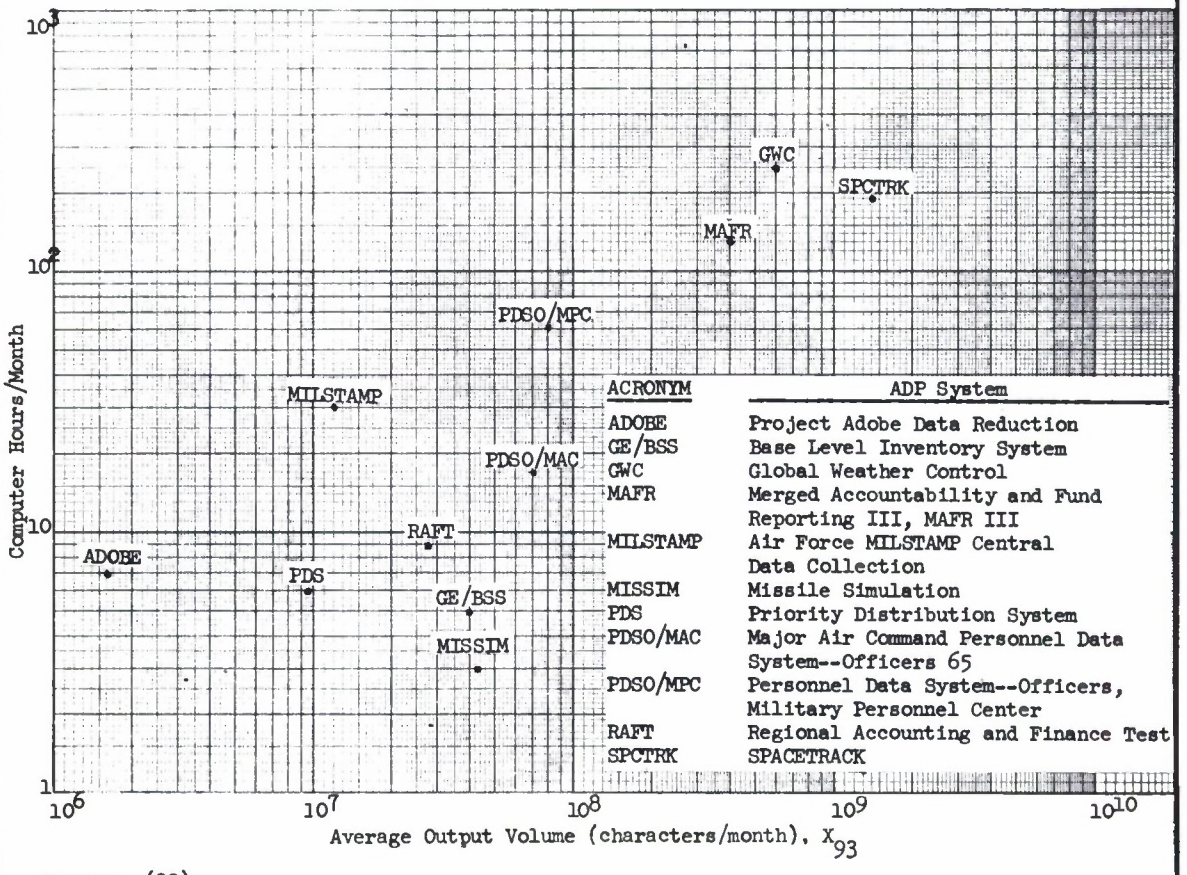


TABLE XLI : PLANNING FACTORS

TYPE: Unit Cost

SUBJECT: Estimating Program Maintenance Computer Hours



Source: (23)

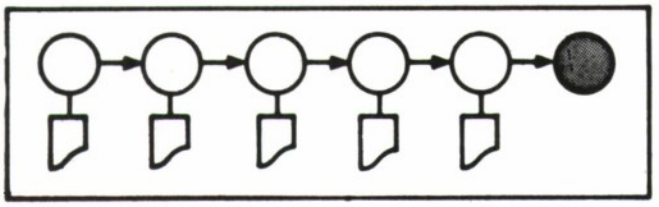
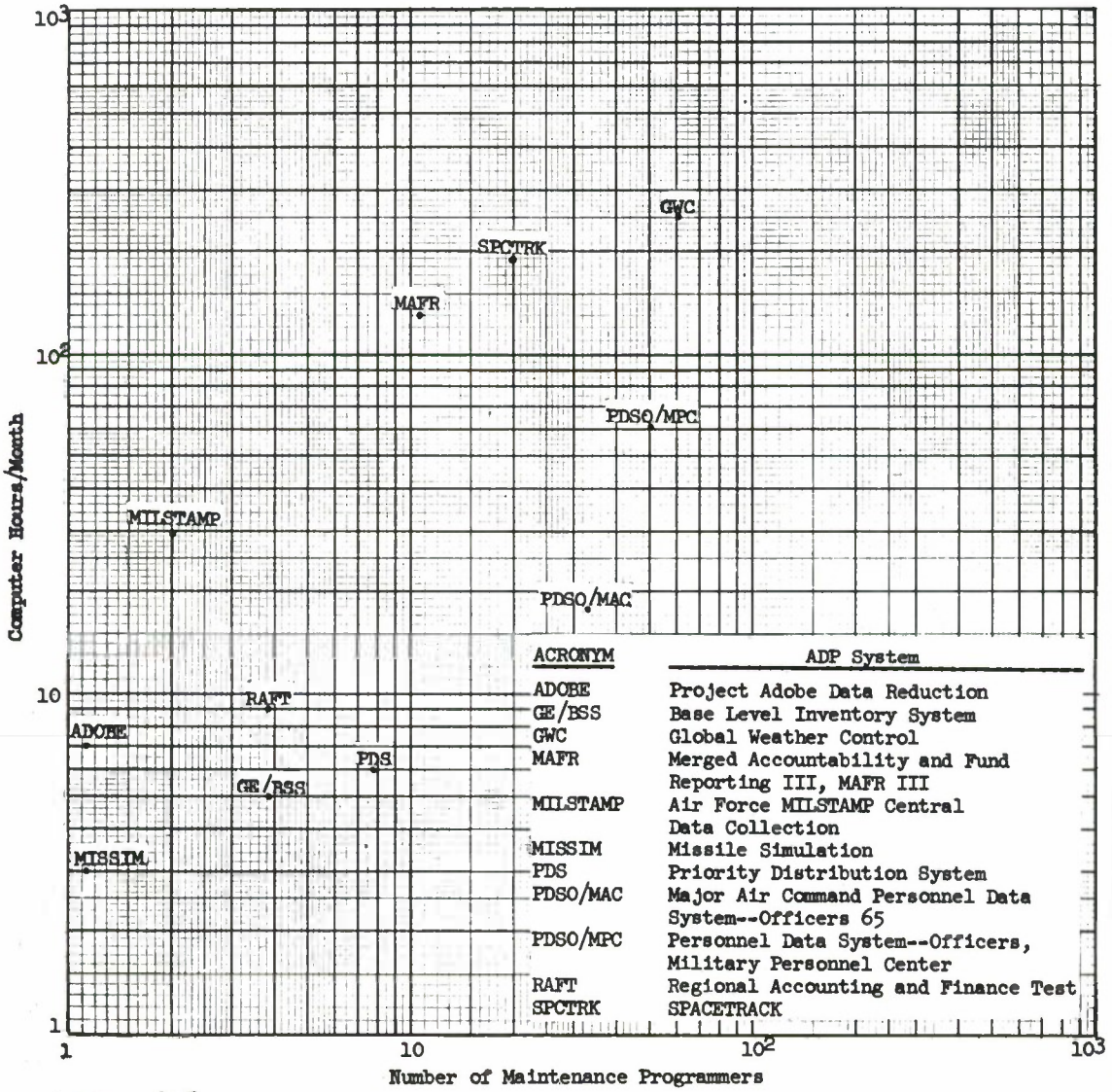


TABLE XLII: PLANNING FACTORS

TYPE: Percent of Other Item

SUBJECT: Estimating Program Maintenance Computer Hours



Source: (23)

GLOSSARY A

DEFINITIONS OF VARIABLES

This glossary contains the definitions for the resource costs and the variables listed in the forms for Computer Programming Cost Factors. There are three kinds of variables defined--first, the dependent variables or resource costs, next, the independent variables or cost factors that originated with the statistical analyses for the Computer Program Design, Code, and Test activity, and then new independent variables to describe further the cost influences on activities in the programming process.

1. Independent Variables (Resources)

Y_1 - Total Number of Man Months including first-line supervision, for the programming step under consideration; does not include the costs of any associated executive or utility program.

Y_2 - Total Number of Computer Hours used by all developmental computers during the programming process step. This includes test time only, not production time on line runs.

Y_3 - Months Elapsed, the number of months elapsed between the start date of the programming process step and the completion date for the step.

2. Dependent Variables (Cost Factors). The variables defined below (ranging from X_1 through X_{77}) were first identified in the statistical analyses that led to the numerical results (Planning Factors and Estimating Equations) in Section V on Computer Program Design, Code, and Test. In some cases, we have generalized the original definitions to extend their use to the other five activities in computer programming and also in some cases the names of the factors have been changed. To help the readers who may have read earlier reports from the Programming Management Project, we have cited reference (20), TM-3026, "Current Results from the Analysis of Cost Data for Computer Programming," 26 July 1966, the most recent Project report, to show the relationship between any of the changed titles and definitions and their earlier forms. The coding used to quantify the variables for the earlier analysis is assumed to be suitable for an initial analysis of the other activities. When one independent variable is related to another, a note has been inserted in the definition.

X_1 - Vagueness of Design Requirements Definition, measures the preciseness of the definition of the programming job. Coded: direct translation of (simple) manual tasks to automatic functions = 0; only a few new functions automated, with requirements relatively clear and precise = 1; many new functions to be automated, requirements relatively clear and precise = 2; many ill-defined and unstructured functions to be automated = 3 (same as "Design Characteristics" in (20).

X_2 - Innovation Required, measures whether a significant portion of the design and production of the program data point involved applications or programming techniques that were new to the personnel involved. Coded: yes = 1; no = 0.

X_3 - Lack of Knowledge of Operational Requirements, measures how well the operational requirements of the program data point were known and documented. Coded: in great detail = 0; in broad outline = 1; only vaguely = 2. See also $X_{45.2}$.

X_4 - Number of Organizational Users, represents the number of organizational interfaces with which the program data point must communicate. This variable has a minimum value of 1.0.

X_5 - Number of ADP Centers, represents the number of computer-based locations with which the program data point must communicate. This variable has a minimum value of 1.0.

X_6 - Complexity of Program System Interface, measures the interaction between the program data point and other programs or I/O equipment. Examples: if the program were a compiler, the measure would represent the degree of programmer effort devoted to data (e.g., source statement) handling, interpretation, and formatting, but not effort spent on internal logic of program; if program involved remote I/O devices, measure would represent the degree of programmer effort devoted to making the data from the I/O devices acceptable to basic program. Coded: more than 50 percent of design effort devoted to data transfer problems to or from the program data point = 2; between 10 percent and 50 percent effort to data transfer problems = 1; less than 10 percent = 0 (same as "Complexity of Communication" in (20).

X_7 - Response Time Requirements, measures the time restraints within which the operating program must perform. Coded: greater than 1 day = 0; 24 hours or less = 1; 1 hour or less = 2; real time = 3.

X_8 - Stability of Design, measures the frequency and extent of program design changes. Coded: initial design carried through without change = 0; few changes to initial program design = 1; frequent changes to program design = 2; initial program design almost completely revised = 3. See also X_7.

X_9 - On-Line Requirements, examines the operational characteristics of the program. Coded: no on-line, real-time operation = 0; mixture of on-line and off-line operations = 1; mainly on-line, real-time operation = 2.

X_{10} - Total Object Instructions Delivered, the total number of object instructions delivered as part of the program data point. This number should include object instructions borrowed and/or reused from other programs, which are actually a part of the total program listing, as well as those instructions specifically written for the program. Coded in thousands.

X_{11} - Percent Delivered Object Instructions Reused, the ratio of reused object instructions to the total number of object instructions delivered.

X_{12} - Total Nondelivered Object Instructions Produced, the total number of object instructions written, but not delivered as part of the finished package. This should include all utility and support programs which had to be written (but not delivered) in order to produce the desired program. Coded in thousands.

X_{13} - Total Source Instructions Written, the total number of source instructions (assembly and procedure-oriented) written. Coded in thousands.

X_{14} - Percent Source Instructions Written in POL, the ratio of procedure-oriented (POL) source language statements written to the total number of source instructions written.

X_{15} - Percent of Total Object Instructions Discarded, the ratio of the number of generated object instructions discarded due to errors and design changes, to the total number of object instructions generated.

X_{16} - Percent of Total Source Instructions Discarded, the ratio of the number of source instructions discarded due to errors and design changes, to the total number of source instructions written.

X_{17} - Number of Conditional Branches, the number of machine language (object code) statements that are conditional branches or jumps. Coded in thousands.

X_{18} - Number of Words in the Data Base, number of words in the subset of tables that describe the environment of the problem to be solved by the program data point and/or files to be processed. Coded in thousands.

X_{19} - Number of Classes of Items in the Data Base, classes of items (i.e., variables) being categories such as names, salaries, states, or any other characteristic of information for which there are many items or entries.

X_{20} - Number of Input Message Types, the number of different records (groups of fields of information treated as a unit) used in input to the program data point.

X_{21} - Number of Output Message Types, the number of different records (groups of fields of information treated as a unit) processed as outputs from the program data point. The number of distinct report formats.

X_{22} - Number of Input Variables, the number of different quantities or fields of information that appear as inputs to the program data point and which assume any value in a set of numbers or symbols.

X_{23} - Number of Output Variables, the number of different quantities or fields of information that appear as outputs from the program data point.

X_{24} - Number of Words in Tables, and Constants not in Data Base, includes all words in tables and constants not in the data base but considered to be describing the problem environment. See variable X_{18}.

*X_{25} - Percent Clerical Instructions, the part of the instructions comprising the program data point that are used for bookkeeping, sorting, searching, and file maintenance.

*X_{26} - Percent Mathematical Instructions, the part of the instructions comprising the program data point that are used for evaluating and computing algebraic, mathematical, geometric, and trigonometric formulas.

---

*Note: Variables X_{25}-X_{29} characterize the computer program by percentage of instructions used for various levels of data processing. The levels are a gross way to indicate types of data processing from a programmer's point of view. Variables X_{30}-X_{35} stem from a different, less precise way to characterize a computer program. Percentage of functions, the basis for this classification, is oriented toward a user's description of a computer program. For example, percent generation (X_{35}) refers to the proportion of program functions devoted to the creation of information from data by the application of some logical process.

*X_{27} - Percent Input/Output Instructions, the part of the instructions comprising the program data point that are used for performing data acceptance and output formatting.

*X_{28} - Percent Logical Control Instructions, the part of the instructions comprising the program data point that are used for the sequencing of operations according to orders, priorities, or timing requirements.

*X_{29} - Percent Self-Checking Instructions, the part of the instructions comprising the program data point that are used for monitoring programs which detect, report, and in some cases attempt to correct errors.

*X_{30} - Percent Information Storage and Retrieval Functions, that part of the program data point devoted to the storage and retrieval of data.

*X_{31} - Percent Data Acquisition and Display Function, that part of the program data point devoted to data acquisition and display procedures.

*X_{32} - Percent Control or Regulation Function, that part of the program data point devoted to the control or regulation of data and process flow.

*X_{33} - Percent Decision-Making Functions, that part of the program data point devoted to logical alternatives and choices in the process flow.

*X_{34} - Percent Transformation Functions, that part of the program data point devoted to the transformation and/or reformatting of data.

*X_{35} - Percent Generation Functions, that part of the program data point devoted to generating the required outputs in the program.

X_{36} - Average Operate Time, the actual average time lapse for processing the average data load with the operational program data point. Coded: not applicable (e.g., utility routines) = 0; less than 15 minutes = 1; 15 minutes to 1 hour = 2; greater than 1 hour = 3.

X_{37} - Frequency of Operation, the average number of times of program data point operation. Coded: not applicable = 0; less than 1/month = 1; more than 1/month and less than 1/week = 2; more than 1/week and less than 1/day = 3; daily = 4; utility or on-line (including compilers) = 5.

---

*See note on previous page.

X_{38} - Insufficient Memory, if a memory size was a factor to be considered in the programming of the problem, enter 1; if no memory constraint existed, enter 0.

X_{39} - Insufficient I/O Capacity, if the lack of I/O channels imposed programming difficulty, enter 1; if not, enter 0.

X_{40} - Stringent Timing Requirements, if strict schedule restrictions were imposed on the program development effort, enter 1; if no tight schedules existed, enter 0.

X_{41} - Number of Subprograms, a subprogram being a well defined set of instructions to perform a mathematical or logical operation within a specific program data point.

X_{42} - Programming Language, i.e., whether the source language consisted of assembly or machine-oriented language instructions or procedure-oriented language statements. Coded MOL = 1; POL = 0.

X_{43} - POL Expansion Ratio, measures the rate of expansion between POL source statements and the corresponding generated machine code.

X_{44} - Support Program Availability, that is, the extent to which support software needed to produce the program data point was available. The extent to which support software had to be written for a given data point is measured by variable X_{12}. In general terms, covering all steps in the programming process, this variable is related to X_{89}.

X_{45} - Internal Documentation, measures the number of pages of documentation distributed within the programming organization, such as program data point specifications, design specifications, etc.

$X_{45.1}$ - Documents written during a programming step.

$X_{45.2}$ - Documents available (extent of documentation) from previous step.

X_{46} - External Documentation, measures the number of pages of documentation written for or distributed to customers, such as program design specifications and operating instructions, during indicated step.

X_{47} - Total Number of Document Types, includes all distinct document types, e.g., flow charts, operating instructions, and program design specifications.

$X_{47.1}$ - Total number of external document types written during a programming step.

$X_{47.2}$ - Total number of internal document types available (i.e., the extent of internal documentation) from previous step.

$X_{47.3}$ - Total number of internal document types written during a programming step.

X_{48} - Type of Program, such as business, scientific, utility, or other application.

$X_{48.1}$ - Business
$X_{48.2}$ - Scientific
$X_{48.3}$ - Utility
$X_{48.4}$ - Other

} mutually exclusive

Note: Variables $X_{48.1}$ through $X_{48.4}$ coded as a mutually exclusive binary variable, i.e., if a program data point is a business program, that application is scored as 1, while the remaining applications are scored as 0. Obviously, there are many possible ways of classifying programs.

$X_{48.5}$ - Stand-Alone. Coded: yes = 1; no = 0.

X_{49} - Compiler or Assembler Used, such as the various versions of COBOL, FAP, FORTRAN, GAP, etc.

X_{50} - Developmental Computer Used, the make and model of the major computer.

X_{51} - First Program on Computer, whether it is a new machine or new to the installation and to the programmers writing the program. Coded: new = 1; not new = 0.

X_{52} - Average Turnaround Time, measures the time lags (in minutes, hours, days) from the time a programmer submits a run to the time it is returned to him. Coded: less than 2 hours = 0; 2 to 11 hours = 1; 12 to 24 hours = 2; greater than 24 hours = 3.

X_{53} - ADP Components Developed Concurrently, hardware components are to be developed concurrently with the program data point. Coded: yes = 1; no = 0.

X_{54} - Special Display Equipment, use of such graphic display as CRT scopes, etc. Coded: yes = 1; no = 0.

X_{55} - Core Capacity, number of words in the main memory of the major computer used in program data point development.

X_{56} - Random Access Device Used, such as drum, disc, etc. Coded: yes = 1; no = 0, if any such equipment was used.

X_{57} - Number of Bits per Word, number of bits per memory word in the major computer used in development.

X_{58} - Memory Access Time, measures the time required to read and restore a memory word. Coded: microseconds x 10.

X_{59} - Machine Add Time, the time required to acquire and execute one fixed-point add instruction. Coded in microseconds.

X_{60} - Computer Cost, monthly rental or equivalent purchase price. Coded: large-scale computer ($750,000 and up) = 0; medium-scale computer ($100,000 to $749,000) = 1; small-scale computer (under $100,000) = 2.

X_{61} - Percent Senior Programmers, the ratio of the total number of senior and systems programmers, to the total number of programmers assigned to the project, based on the following position descriptions:

Position	Description
Coder	Writes machine language instructions from flow charts. Helps prepare flow charts and test programs.
Programmer	Develops programs to solve well-defined problems. Prepares flow charts, writes instructions, tests programs, modifies established computer programs.
Senior Programmer	Conceives, develops and improves large, complex computer programs, e.g., automatic programming routines. Improves efficiency of existing programs.
System Programmer	Formulates and plans new program system applications. Keeps abreast of related economic disciplines and new information processing technology. Is highly creative in designing and developing major computer program systems.

X_{62} - Average Programmer Experience with Language, the average number of months of experience that the assigned programming staff has had with the language used in coding the program data point.

X_{63} - Average Programmer Experience with Application, the average number of years' experience that the assigned programming staff has had with the application represented by the program data point.

X_{64} - Percent Programmers Participating in Program Design, the ratio of programmers participating in the design of the program data point to the total number of programmers assigned to the program data point development.

X_{65} - Personnel Continuity, the number of personnel working for the duration of the project divided by the maximum number assigned at any one time. This variable measures the degree of fluctuation in the size of staff; it is related to X_{92}.

X_{66} - Maximum Number of Programmers, the maximum number of programmers assigned to the program data point at any one time.

X_{67} - Lack of Management Procedures, measures the use of eleven management procedures, such as the use of PERT charts, evaluation of program design changes, dissemination of error-detection and -correction procedures, contingency plans for computer overload, etc. Coded: the number of "no" replies to the eleven procedures (see reference (40)).

X_{68} - Number of Agencies Concurring in Design, the number of different organizations or agencies that have to sign-off on proposed program data point design.

X_{69} - Customer Inexperience, measures customer knowledge and experience in developing EDP systems. Coded: extensive = 0; limited = 1; none = 2.

X_{70} - Computer Operated by Agency Other than Program Developer. Coded: yes = 1; no = 0.

X_{71} - Program Developed at Site other than the Operational Installation. Coded: yes = 1; no = 0.

X_{72} - Different Computers for Programming and Operation, refers to whether a different model of computer was used for program development and operation. Coded: yes = 1; no = 0.

X_{73} - Closed or Open Shop Operation, indicates the type of installation in which the program data point was developed. Coded: open shop = 0; closed shop = 1.

X_{74} - Number of Locations for Program Data Point Development, measures the number of different geographical locations at which the program data point was developed.

X_{75} - Number of Man Trips, measures the number of man trips required for concurrence during design, code, and test phases of program data point development.

X_{76} - Program Data Point Developed by Military Organization. Coded: yes = 1; no = 0.

X_{77} - Program Data Point Developed on Time-Shared Computer. Coded: yes = 1; no = 0.

Note: Variables X_{78} through X_{91} are factors that have not been used in any numerical analysis. Therefore, they represent hypotheses which have not been investigated in terms of data; they may not be as precisely defined as the previous 77 factors.

X_{78} - Complexity of System Interface with Other Systems, would measure the dependence of the subject system, i.e., that under development, upon other data processing (information) systems in terms of providing inputs to them or accepting outputs from them. The complexity would be a function of the number of messages from or to each of the related systems, as well as the degree of dependence as it relates to performance of specific organizational missions, for example, messages to and from a computer in a missile for guidance and control would represent a high degree of dependence in executing an organizational mission. Related to X_6.

X_{79} - Security Classification Level, would measure security classification level (e.g., Company Private, Confidential, Secret, Top Secret). The hypothesis is that the higher the level of classification, the more costly (and time-consuming) the work in terms of obtaining personnel and documents containing essential information, safeguarding the current results of the work and transmitting these results to other cognizant personnel.

X_{80} - Number of Sources of System Information, would measure the dispersion of various kinds of information needed to conduct the system analysis and design work, as well as other activities in computer program development. The hypothesis is that if there are a large number of sources, then the cost of the work would increase; for example, interviewing a large number of personnel or securing documents from different libraries or information sources.

X_{81} - Accessibility of System Information, is closely related to X_{80}, that is a measure of the effort with which information may be acquired. (In this sense, it is also related to X_{79}, security classification.) The hypothesis here is that if information, such as documents, is easy to obtain, then the cost of system analysis and design work as well as the conduct of other activities in computer programming would be reduced.

X_{82} - Degree of System Change Expected During Development, would measure the expected change, for example, in interfacing systems and in organizational mission reflected in the system functions during the development period. The hypothesis is that the greater the amount of change expected, the higher the cost will be during the entire process of computer program development, particularly during system analysis and design; this is because effort is spent making allowances for contingencies.

X_{83} - Degree of System Change Expected During System Operations. Closely related to X_{82}, this factor would be a measure of the amount of flexibility and versatility that must be designed into the proposed system. Therefore, the hypothesis is that additional cost would be incurred with a higher degree of expected change and would be reflected throughout the computer programming process.

X_{84} - Number of Functions in the System, would measure the specific parts of the organizational mission that would be changed by the development of the new system which would provide automatic data processing for all or parts of these functions. The hypothesis is that the greater the number of functions, the higher the cost in all of the computer program development activities.

X_{85} - Number of System Components, a count of the various system elements that are part of information processing or controlled by it. For example, sensors, communication links, communication terminals, weapons, production levels. The hypothesis is that the greater the number of components, the greater the cost.

X_{86} - Number of System Components--Not Off-the-Shelf, measures the newness or innovation required in all (ADP and other) system components. It is closely related to X_2, X_{53}, and X_{85}. The hypothesis is that as the number of components that must be developed along with computer programs increases so does the cost of computer program development.

X_{87} - Percent Senior Analysts (related to X_{61}, percent senior programmers), is a measure of the extent to which experience and proficiency are applied to the job. One hypothesis is that the higher the cost of the initial investment, particularly in system design and analysis, the lower the total cost of the system over its lifetime. In the systems analysis step, the hypothesis is that more proficient analysts do the job with fewer resource units.

X_{88} - Quality of Resource Documents, would be a measure of the accuracy consistency, completeness, freshness (up to date), and ease of understanding for the documents used as sources of information in doing computer program development, particularly the system analysis and design work. The hypothesis is that the better the quality, the lower the cost.

X_{89} - The Availability of Special Tools; for example, computer programs for simulation. This factor is a measure of the accessibility and reliability of tools that could be used to facilitate the computer program development, particularly system analysis and design. The hypothesis is that if such tools are available, they will help speed the work and reduce the cost. This factor is related to, but is an expansion of, X_{44}.

X_{90} - Degree of Standardization in Policy and Procedures, measures how how much formal guidance has been introduced for controlling the process of computer program development (possibly as part of the development of a larger system). The hypothesis is that the more such standards are introduced, the higher the cost will be for the initial development process (since these actually are a kind of constraint), but the lower the cost of the total system over its lifetime.

X_{91} - Number of Official Reviews of Documents, may be a direct count of the formal inspections during the process of computer program development. The hypothesis is that the greater the number of these reviews, the higher the cost of computer programming; but the lower the total cost of system maintenance over its lifetime. This factor is related to X_{68}.

X_{92} - Personnel Turnover, the number of personnel on the project replaced per time period, divided by the total staff. Turnover measures the instability of the staff assigned to the project. It is related to X_{65}; but X_{92} refers to the replacement of individual project members.

X_{93} - Output Volume, the expected amount of output from the user's operation of the program data point. Measured in average characters, exclusive of blanks, per month.

X_{94} - Input Volume, the expected amount of input for user's operation of the program data point. Measured in average number of characters per month.

GLOSSARY B

DEFINITION OF SYMBOLS

This glossary contains the definitions of symbols that occur in the text--mainly in the Planning Factors and Estimating Equations in Section V, Computer Program Design, Code, and Test.

r = multiple correlation coefficient
This statistic is a measure of the interrelationship among all the independent variables, (cost factors), and the dependent variable (cost)

r^2 = coefficient of determination

$$= \frac{\text{explained sum of squares}}{\text{total sum of squares}}$$

This statistic, the square of the multiple correlation coefficient shows how much of the variation in the dependent variable being predicted is explained by the estimating relationship, e.g., an equation. The range is zero to one; the higher the value the better the equation.

N = total number of data points in a given sample

β = standardized regression coefficient (3)
This statistic is a measure of the relative strength of a particular independent variable (cost factor) that appears in an equation

σ = standard deviation of a variable
This statistic is a measure of the spread or variation in a variable. In this Handbook, the standard deviation is usually shown for a cost or dependent variable.

σ_E = standard error of the estimate of a regression equation
This statistic, a measure of the estimating precision (accuracy) of the estimating equation, determines the confidence intervals such as the Stanine bands used in the text. The smaller the value the better the equation.

GLOSSARY C

USE OF TERMS

This glossary describes the ways particular terms, usually familiar words, are used in this Handbook.

1. Program Data Point. Each member of the sample of computer programs representing a particular program for which separate and distinct data are available. To qualify as a data point, the data must be for a programming effort that resulted in (a) the smallest number of instructions developed for a user with specific computer programming requirements, and (b) a program or product capable of operating in the computer as a single entity or package.

2. Open Shop. A computer facility with operating procedures that allow programmers to have direct access to the computer for the purpose of running and debugging programs.

3. Closed Shop. A computer facility with operating procedures that require programmers to submit all jobs to a second party. The programmer has no control over the scheduling of jobs, or the physical operation of the hardware.

4. Resource Unit. A single quantity of any of the resources used in the computer programming process. The three resource units considered in this report are the man month, the computer hour, and the month of elapsed time.

5. Computer Program. The words computer program may mean several different things, depending upon the context, e.g., a computer program system that consists of an interrelated collection of programs designed to work together to perform certain data processing functions, a part of such a program system corresponding to a single function, or a single unit that performs by itself and corresponds to a single function. Throughout the text, qualifiers have been used to establish a context that helps the reader interpret the meaning appropriately. Also, in some places, for clarity, synonyms are used for the words computer program when they refer to part of a computer program system, e.g., subprogram, run, individual program, routine.

6. System. This term is used to refer to either a computer program system or an information (processing) system. Qualifiers are used in context to help the reader interpret the term correctly. For example, in the sentence "...determining...requirements for improved information processing and planning of a system plus a set of computer programs...." the system is an information processing system.

7. Information System. This term, a short form of information processing system, is used to put computer programming work in a context. In its broadest sense, an information system includes men, machines, procedures, communications, and the computer program that work together at one or more locations to process information automatically and manually to help fulfill one or more missions in an organization. The automatic parts of an information system could be called an automatic data processing system.

8. Manual. This term, used as an adjective, means nonautomatic.

9. Data Processing. This terms is used as a synonym for automatic data processing.

10. Data Processing Project. This term refers to a specific organized effort to develop (or modify) a data processing system and is sometimes used synonymously with data processing system. The development work for a data processing project includes computer programming as one major type of work.

11. Function. This term refers to an integral unit of data processing that corresponds to a user's information processing responsibility and that is usually described in terms understandable to both user and information processing personnel.

GLOSSARY D

DEFINITION FOR THE CODE TO RATE THE IMPACT OF COST FACTORS

This glossary contains definitions of the code used to rate the influence or impact of a particular factor upon a particular resource cost. In the text, these ratings are found in the forms for computer programming cost factors in each of the sections (III through VIII) on the programming activities. Most variables (cost factors) listed for each step in the programming process will have a plus (+) or minus (-) sign to indicate the presumed direction of impact of the variable on the indicated resource. In addition, the numeric indicator for correlation could be determined only when empirical data were available; these indicators appear only in the section on Computer Program Design, Code, and Test. Also, some cost factors in this section were given an additional rating to explain why the variable is included.

1. Correlation. The following numerical ratings indicate the degree of correlation of the independent variable with the indicated resource:

4 = High correlation: statistically significant at the 1% level ($r \geq .20$ for $N = 169$)

3 = Moderate correlation: statistically significant at the 5% level ($.20 > r \geq .15$ for $N = 169$)

2 = Low correlation ($r < .15$)

1 = Indeterminate: present data base not adequate for statement of degree of correlations for this variable; or correlation has an illogical sign for some subsamples.

Reference 6 contains the specific correlation coefficients for the variables rated numerically in this Handbook. In general, the definitions for the variables that appear in Glossary A were so constructed that the sign of the correlation coefficient would be expected to be positive. Thus, X_{60}--Computer Cost, was coded 0 for large-, 1 for medium-, and 2 for small-scale computers, since an increase in the variable so coded would probably result in an increase in the resource under consideration.

2. Signs. The direction of relationship is indicated by:

+ = Resource varies, or should be expected to vary, directly with variable; an increase in the value of the variable, as defined in Glossary A, results in an increase in the amount of the resource.

- = Resource varies, or should be expected to vary, inversely with variable; an increase in the value of the variable as defined in Glossary A, results in a decrease in the amount of the resource.

The lack of a sign code indicates that the direction of variation of the resource with the presence of the factor is not clear, or should not be presumed. Sign codes will always be present with correlation codes 4, 3, or 2; but sign codes may be absent when the correlation code is 1. When only a sign code appears, the variable has not been analyzed statistically.

3. Other Considerations. In addition, the following rating codes appear only for cost factors in the design, code, and test step in the programming process:

A = This cost factor, taken independently, has high correlation with the indicated resource; but this factor does not appear in the estimating equations, because it has been replaced by another superior variable with which it is closely related, i.e., correlates.

B = Analysis to date has not demonstrated the impact of this variable for the indicated resource. However, the variable maintains strong intuitive appeal, and merits further research.

REFERENCES

1. Air Force Systems Command. "Configuration Management During the Definition and Acquisition Phases," AFSCM 375-1, 1 June 1964; "System Program Office Manual," AFSCM 375-3, 15 June 1964; "Systems Program Management Procedures," AFSCM 375-4, 15 June 1965; "System Engineering Management Practice," AFSCM 375-5, 10 March 1966; "Development Engineering," AFSCM 375-6, June 1964.

2. Air Force Systems Command. "Management of Contractor Data and Reports." AFSCM 310-1B, Volume II, 1 January 1960.

3. Anderson, R. L., and T. A. Bancroft. Statistical Theory in Research. New York, McGraw-Hill, 1952, pages 168-190.

4. Anthony, Robert N. Management Accounting. Richard D. Irwin, Inc., Homewood, Illinois, 1960, pages 488-490, 532-540.

5. Auerbach Standard EDP Reports. Auerbach Corporation, Philadelphia, Pennsylvania.

6. Benson, S., G. Neil, and L. Searle. "Supplementary ESD Narrative Material for AFSC 375-4." System Development Corporation, TM-2661 (Draft), 29 September 1965, 64 pages.

7. Boutell, Wayne S. "Problem-Oriented Languages: FORTRAN vs. COBOL," Management Services, May-June 1966, pages 41 and 46.

8. Brandon, Dick H. Management Standards for Data Processing, D. Van Nostrand Company, Inc., Princeton, New Jersey, 1963.

9. Bureau of the Budget. Report to the President on the Management of Automatic Data Processing in the Federal Government, 89th Congress, 1st Session. U. S. Government Printing Office, Washington, 1965, pages 27-40.

10. "The California Computer Circuit," in Datamation, December 1962, page 36.

11. Canning, Richard G., Editor. "New Ways for EDP System Studies," EDP Analyzer, Vol. 1, No. 8, September 1963, pages 9 and 10.

12. Carlson, Gary. "Predicting Clerical Error," Datamation, February 1963, pages 34 and 35.

13. Chapin, George G., and Paul A. Hensel. Programming for the Naval Tactical Data System. UNIVAC Division of Sperry Rand Corporation, St. Paul, Minnesota, pages 15-16.

14. Cowan, Royden A. "Is COBOL Getting Any Cheaper?" Datamation, Vol. 10, No. 6, June 1964, pages 46-50.

15. Dalleck, Winston C. "Computers, Models, and Business Management," EDP: The First Ten Years, McKinsey & Company, Inc., page 34.

16. Durost, W. N., The Characteristics, Use, and Computation of Stanines. Test Service Notebook No. 23, Harcourt, Brace and World, Inc., 1959.

17. "Eighth Annual Report on EDP Salaries," Business Automation, June 1966, pages 36-47.

18. Farr, L., V. LaBolle, and N. E. Willmorth. "Planning Guide for Computer Program Development." System Development Corporation, TM-2314/000/00, 10 May 1965.

19. Farr, L., and H. J. Zagorski. "Factors that Affect the Cost of Computer Programming: A Quantitative Analysis." System Development Corporation, TM-1447/001/00, 31 August 1964, pages 59-86.

20. Fleishman, T. "Current Results from the Analysis of Cost Data for Computer Programming." System Development Corporation, TM-3026/000/01, 27 June 1966, pages 35-85.

21. Frank, W. L., W. H. Gardner, and G. L. Stock. "Programming On-Line System," Datamation, June 1963, pages 28-29.

22. Gallagher, James D., and Douglas J. Axsmith. "Data Processing in Transition," EDP: The First Ten Years, McKinsey & Company, Inc., pages 10-11.

23. Gradwahl, Alan J., et al. Phase II Midpoint Report on Use of Air Force ADP Experience in Judging Proposals for New Automation, Planning Research Corporation, July 1966.

24. Haverty, J. P. "Programming Language Selection for Command and Control Applications."* The RAND Corporation, September 1964, pages 4 and 14.

25. Hosier, W. A. "Pitfalls and Safeguards in Real-Time Digital Systems," Datamation, May 1962, pages 68-72.

26. "IBM's $5,000,000,000 Gamble, " Fortune, September 1966, pages 118-123, 224-228. Continued as "The Rocky Road to the Marketplace," Fortune, October 1966, 138-143, 199-212; see especially pages 139, 212.

27. Israel, David R. System Design and Engineering for Real-Time Military Data Processing Systems. The MITRE Corporation, SR-124, September 1964, pages 51-54.

---

*Prepared for presentation at the Symposium on Computer Programming for Military Systems, SHAPE Technical Centre, The Hague, The Netherlands, September 28-October 2, 1964.

28. Johnston, J. Statistical Cost Analysis. McGraw-Hill Book Company, Inc., New York, 1960.

29. Jones, M. V. "How to Estimate the Investment Cost of Electronic Data Processing Equipment." The MITRE Corporation, MTR-70, May 1966.

30. Keese, W. M., et al. "Management Procedures in Computer Programming for Apollo--Interim Report." Bellcomm, Inc., TR-64-222-1, 30 November 1964.

31. LaBolle, Victor. "Development of Equations for Estimating the Costs of Computer Program Production." System Development Corporation, TM-2918/000/00, 5 April 1966.

32. LaBolle, Victor. "Management Aspects of Computer Programming for Command and Control Systems." System Development Corporation, SP-1000/000/02, 5 February 1963.

33. "Last of the Breed," News Item, Electronics, 19 September 1966, page 201.

34. Lawson, Charles M. "Computing Facility Management Survey Results and Analysis." System Development Corporation, TM-704, 19 March 1962. A summary of this survey appears in Datamation, July 1962.

35. Mansfield, J. C. "Personnel Research at SDC: Reports and Papers." System Development Corporation, TM-2902/010/00, 1 April 1966.

36. McElwee, D. W., and J. E. Fernandes. A Software Primer for Managers. Industrial College of the Armed Forces, Washington, D. C. April 1964.

37. Naftaly, Stanley M. "Correcting Obfuscations by Ordained Linguists," Datamation, May 1963, pages 24, 26, 27, 28.

38. Nelson, E. A. "Management Theory and the Economic Size of Organizations." System Development Corporation, SP-2605, 11 October 1966.

39. Nelson, E. A. "Methods of Obtaining Estimates of Computer Programming Costs: A Taxonomy." System Development Corporation, TM-3105/000/00, 15 August 1966.

40. Nelson, E. A. "Research Into the Management of Computer Programming: Some Characteristics of Programming Cost Data from Government and Industry." System Development Corporation, TM-2704/000/00, 15 November 1965, pages 18-43.

41. Novick, David, Editor. "Program Budgeting." The RAND Corporation, Superintendent of Documents, U. S. Government Printing Office, 1965.

42. Novick, David. "System and Total Force Cost Analysis." The RAND Corporation, RM-2695, 15 April 1961.

43. "One Compiler, Coming Up!" Datamation, May-June 1959, page 15.

44. Parker, R. W. "The SABRE System," Datamation, September 1965, page 52.

45. Patrick, Robert L. "The American Way," Datamation, May 1963, page 65.

46. Perry, D., and G. Cantley. "Computer Programmer Selection and Training in System Development Corporation." System Development Corporation, TM-2234, 2 February 1965.

47. Philipson, H. L., Jr. "Optical Character Recognition: The Input Answer," Data Processing, Vol. X, Proceedings of the 1966 International Data Processing Conference, Data Processing Management Association, June 21-24, 1966. Mr. Philipson is resident of Recognition Equipment, Incorporated.

48. Pridmore, H. D. "Organizing for Nationwide DP--Australian Census Network," Datamation, March 1965, page 29.

49. Proceedings of the Second Annual Conference, Computer Personnel Research Group, TG-619, held at New York University, 20-21 July 1964. The Johns Hopkins University/Applied Physics Laboratory.

50. "The RAND Symposium: 1962," Datamation, October 1962, pages 27 and 32.

51. Ross, Duane G. "Input Conversion Processes and Considerations." System Development Corporation, TM-LO-2002/000/00, 7 October 1965.

52. Schmidt, D. T., and T. F. Kavanagh. "Using Decision Structure Tables," Datamation, March 1964, pages 48 and 54.

53. Searle, L. V., and G. Neil. "Configuration Management of Computer Programs for Information Systems." System Development Corporation, TM-1918/000/01, 12 July 1965.

54. Sharpe, W. F. "Financing Electronic Data Processing Equipment in the Federal Government: A Comparison of Purchase and Leasing," Quarterly Review of Economics and Business, Vol. 3, No. 4, Winter 1963, pages 59-66.

55. Shaw, C. J. "Assemble or Compile?" Datamation, September 1966, pages 59-62.

56. Shaw, D. C., and G. B. Davis. "Electronic Data Processing in an Accounting Practice." System Development Corporation, TM-LO-2102/100/00, 31 August 1966, page 13.

57. Snyder, Robert G. "Programming Documentation," Datamation, October 1965, pages 44 and 48.

58. Steel, T. B., Jr. "Programmers and Cheap Computing," Datamation, August 1965, pages 63 and 65.

59. Steel, T. B., Jr. "UNCOL: Universal Computer Oriented Language Revisited," Datamation, January-February 1960, page 18.

60. Wattenberg, W. H. "The Programming Problem in Command and Control," Datamation, September 1962, page 44.

61. Weinwurm, G. F. "Data Elements for a Cost Reporting System for Computer Program Development." System Development Corporation, TM-2934/000/02, 6 July 1966.

62. Weinwurm, G. F., and H. J. Zagorski. "Research Into the Management of Computer Programming: A Transitional Analysis of Cost Estimation Techniques." System Development Corporation, TM-2712/000/00, 12 November 1965.

63. Willmorth, N. E. "System Program Management: Program and Document Maintenance." System Development Corporation, TM-(L)-2222/018/00, 30 July 1955.

LETTER TO THE AUTHORS

I work in an organization that

- buys or evaluates computer programs ☐
- develops computer programs ☐
- other (state) _____

My main activity is in

- systems analysis ☐
- computer operation ☐
- other programming activities (state) _____

I do work in management Yes ☐ No ☐

if yes:

first-level supervision ☐ higher-level supervision ☐ staff ☐

I rate the Handbook as follows:

	Very Good	Fair	Poor
overall value	____	____	____
overall quality	____	____	____
usefulness	____	____	____
scope	____	____	____
format and organization	____	____	____

I suggest the following improvements in the Handbook:

_____

_____

_____

_____

_____

I suggest the following additions to the Handbook:

_____

_____

_____

_____

_____

I would like to have the following additions to planning factors and equations in the Handbook:

_____

_____

_____

_____

_____

I have the following additional rules of thumb, planning factors, and/or estimating equations used in my organization or elsewhere (please give any references):

_____

_____

_____

_____

_____

Name: _____

Organization: _____

Address: _____

_____

Date: _____